

SAP PRESS

Schrödinger programmiert ABAP

Das etwas andere Fachbuch

Bearbeitet von
Roland Schwaiger

Neuausgabe 2013. Taschenbuch. 735 S. Paperback

ISBN 978 3 8362 1858 0

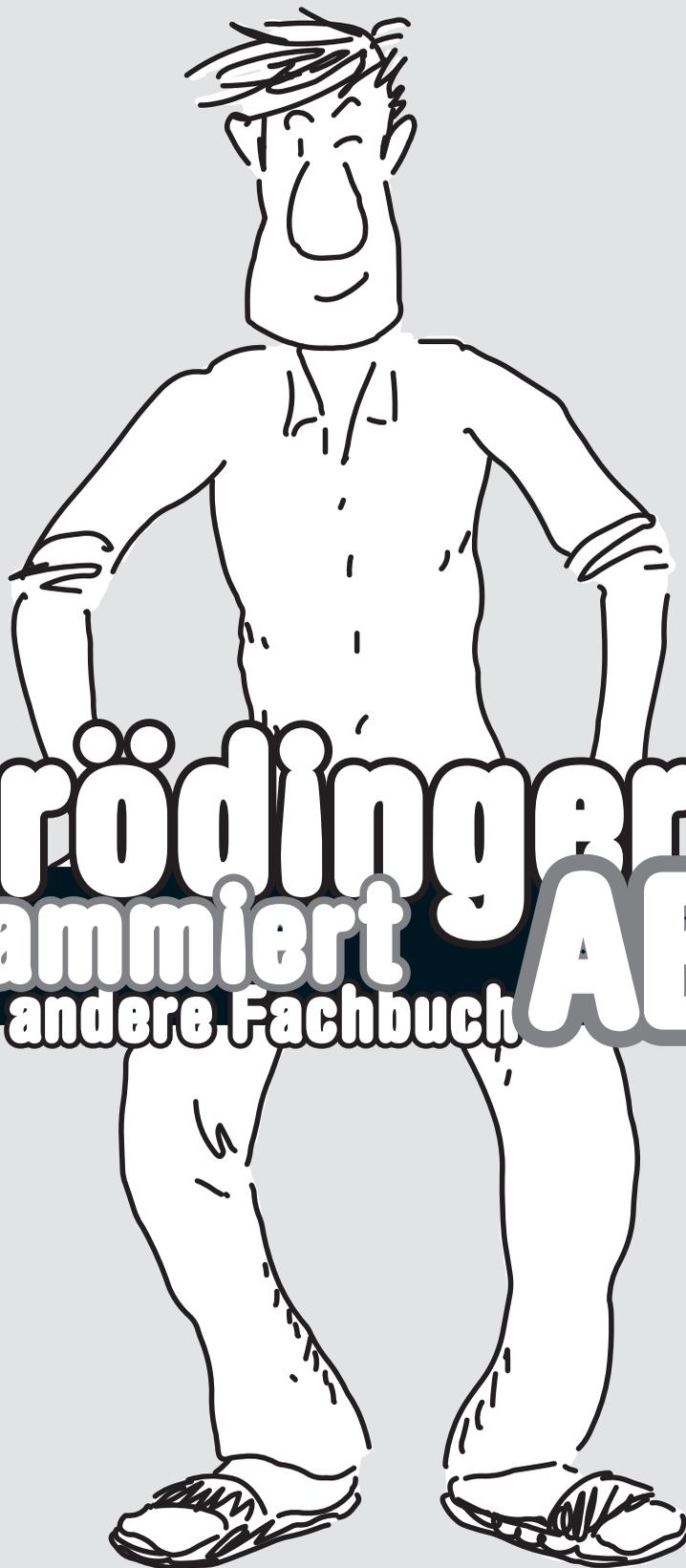
Format (B x L): 20 x 23 cm

[Weitere Fachgebiete > EDV | Informatik > Programmiersprachen: Methoden > Objektorientierte Programmierung](#)

schnell und portofrei erhältlich bei


DIE FACHBUCHHANDLUNG

Die Online-Fachbuchhandlung beck-shop.de ist spezialisiert auf Fachbücher, insbesondere Recht, Steuern und Wirtschaft. Im Sortiment finden Sie alle Medien (Bücher, Zeitschriften, CDs, eBooks, etc.) aller Verlage. Ergänzt wird das Programm durch Services wie Neuerscheinungsdienst oder Zusammenstellungen von Büchern zu Sonderpreisen. Der Shop führt mehr als 8 Millionen Produkte.



Schrödinger programmiert **ABAP**

Das etwas andere Fachbuch

Liebe(r) Leser(in),

oder Ihr Arbeitgeber

Sie haben gewählt:



ABAP

Wir gratulieren – und bedienen Sie gern. Wo soll's denn hin?
Ins Hirn? – Gerne doch.
Die Windungen sehen gut aus, da lässt sich Fachwissen unterbringen.

JA, BITTE.
AM LIEBSTEM IM
SCHLAF. ODER
EINFACH MIT'NEM
TRICHTER...

Apropos unterbringen:

Haben Sie schon mal versucht, eine Vorratskammer mit ganz glatten Wänden zu füllen? – Geht nicht. Nicht ohne Regale jedenfalls. Bretter, Nischen, Vorratsdosen – damit geht's.

Was diesen **Trichter** angeht, da müssen wir leider **passen**. Aber seien Sie nicht allzu enttäuscht, denn Sie halten mehr als einen Ersatz in den Händen:

**Das etwas
andere Fachbuch.**



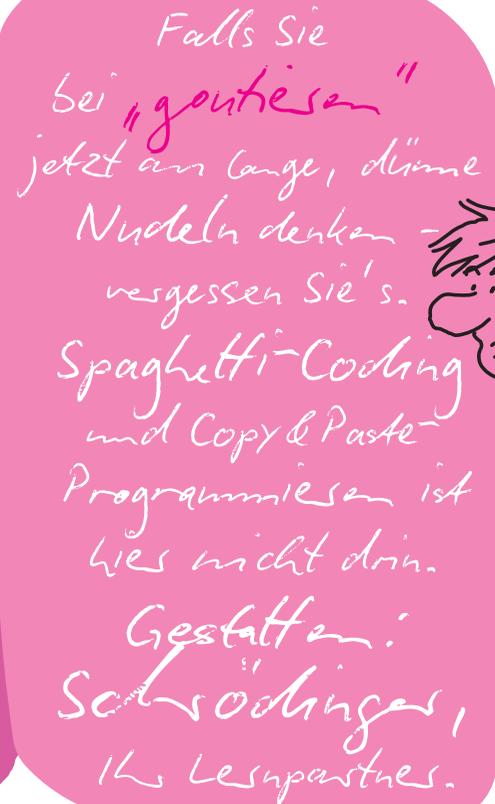


Mit Nischen, Behältern in allen Größen und belastbaren Fachböden. Wir haben keine Mühe gescheut.

Wir haben einen hervorragenden Ausbilder engagiert.

Ein Experten-Team hat die Wände behauen, den Code eingefärbt und die nötigen Umwege eingebaut, so dass Sie bequem goutieren können.

Wobei **bequem** nicht „passiv“ bedeutet, und **goutieren** nicht „schlafen“, aber das wäre ja ohnehin viel zu schade.



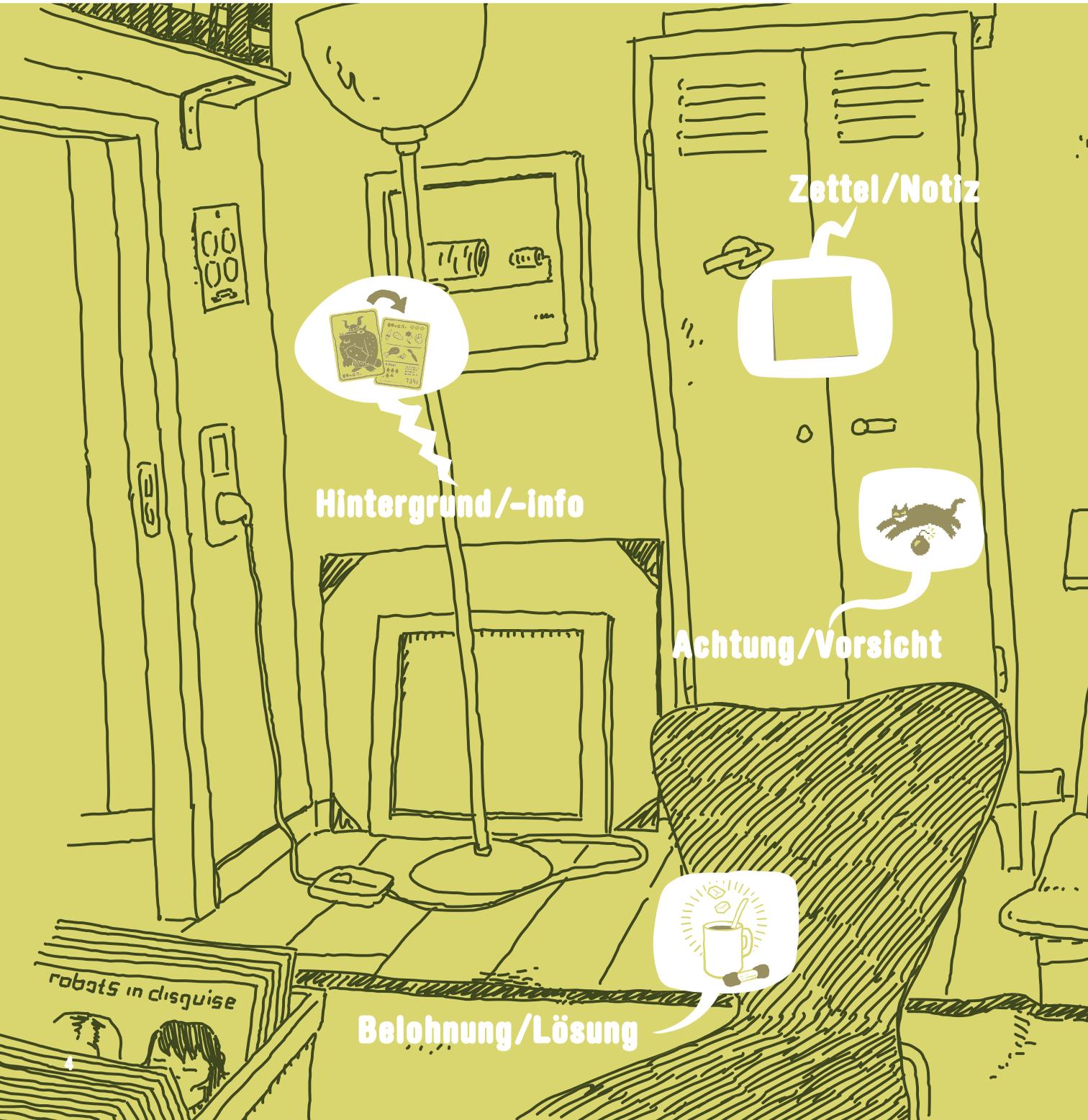
Falls Sie bei „goutieren“ jetzt an Länge, dünne Nudeln denken – vergessen Sie's. Spaghetti-Coding und Copy & Paste-Programmieren ist hier nicht drin. Gestatten: Schrödingers, Ihr Lesepartner.

Bitte sehr. **Es ist angerichtet.**

Viel Spaß!

Ihr Verlag

Schrödingers Büro



Zettel/Notiz

Hintergrund/-info

Achtung/Vorsicht

Belohnung/Lösung

robots in disguise

Die nötige Theorie, viele Hinweise und Tipps

15:20

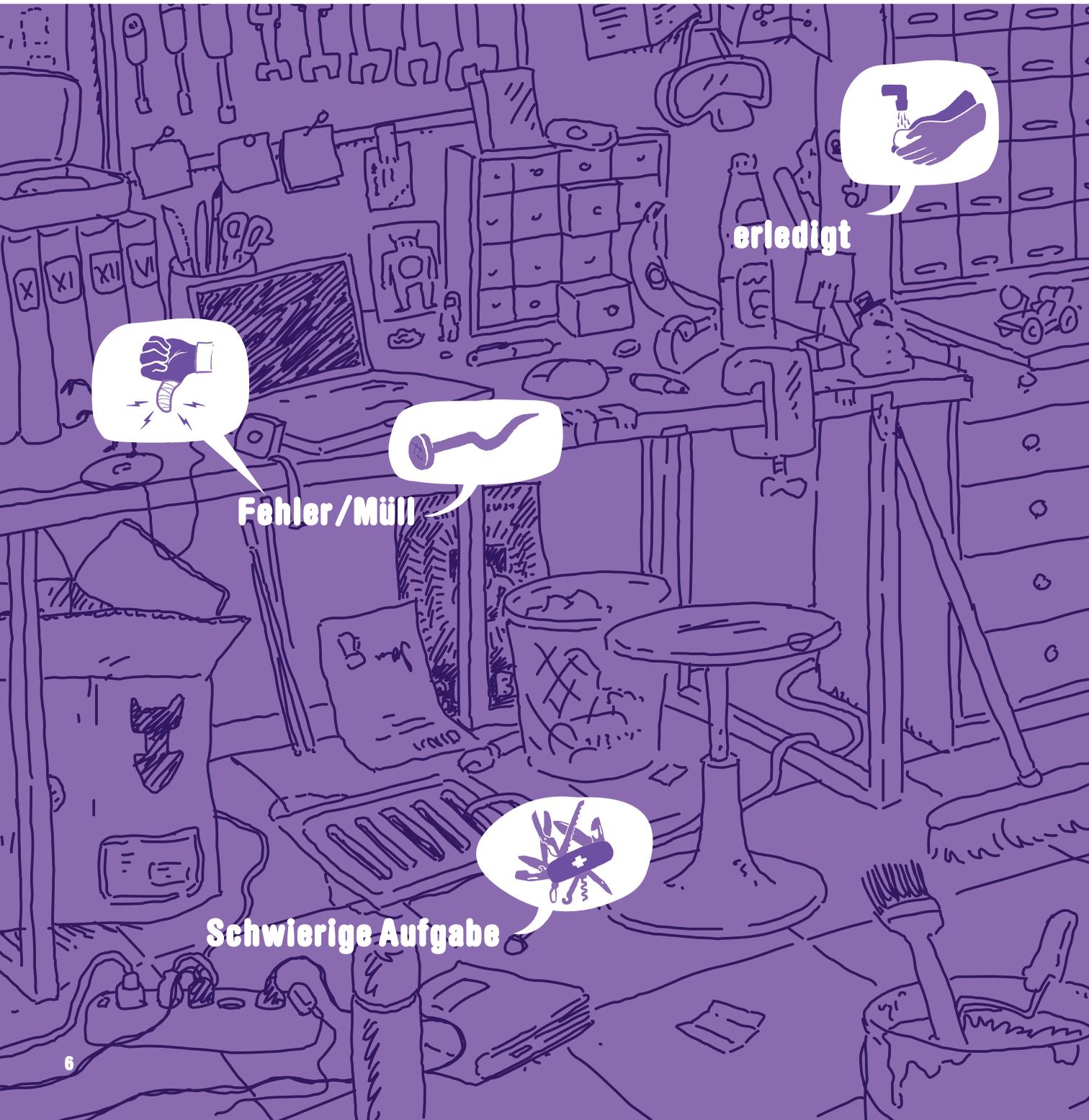
Begriffsdefinition



Ablage



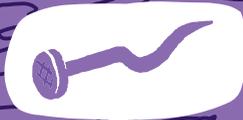
Schrödingers Werkstatt



erledigt



Fehler / Müll



Schwierige Aufgabe

Ummengen von Code, der ergänzt, verbessert und repariert werden will



Notiz



CODE bearbeiten/nachbessern



Achtung/Vorsicht



Einfache Aufgabe

Schrödingers Wohnzimmer

Zettel



Belohnung/Lösung



Übung



Mit viel Kaffee, Übungen und den verdienten Pausen



schwierige Aufgabe



einfache Aufgabe



Achtung/Vorsicht



INHALTSVERZEICHNIS

Vorwort 24

Kapitel 1: Eine Nudelmaschine für zu Hause

Die SAP, das SAP-System und ABAP

Seite 27

Den Schrödinger hat es mal wieder erwischt: Ein österreichischer Nudelhersteller hat seine alte Firma, Spaghetti Infinito, übernommen. Dabei war das doch bis jetzt so gemütlich in der Spaghetti-Informatik. Man konnte programmieren, wie und womit man wollte – Hauptsache, es lief. Wenn er zukünftig keinen Spaghetticode mehr programmieren darf, kann er sich gleich etwas Neues suchen: Zum Beispiel bei der ABAP-Fabrik, von der er schon so viel gehört hat. Klingt lustig, ABAP. Und wenn die auch Code produzieren ..., vielleicht kann er dann ja doch wieder ein bisschen Spaghetticode programmieren! An seinem ersten Tag lernt er den Schwaiger Roland kennen, seinen ABAP-Ausbilder. Der seine Vorliebe für Spaghetticode sofort entdeckt und sich zunutze macht. Aber anders, als Schrödinger denkt.

Servus Schrödinger!!!	28	SAP – SAP-System	38
Schrödingers SAP-System – Eigenes		Die Erfolgsgeschichte: Vom ABAP zum ABAP	41
SAP-System runterladen	31	Kompilieren/Interpretieren	45
Schrödingers SAP-System – Das brauchst du dafür	33	ABAP allgemein	46
Schrödingers SAP-System – Installieren	34	ABAP-Syntax	47
Die SAP – Das Unternehmen	37	Beispiel ABAP	49
		Schrödingers ABAP	50

Kapitel 2: Rein in die Küche – raus aus der Küche

Starten mit dem SAP-System

Seite 51

Ein spannender Tag für den Schrödinger: Das erste Mal! Der Schwaiger Roland meint, das tut gar nicht so weh, wie alle sagen, der hat leicht reden. Und fängt gleich mit der Anmeldung am SAP-System an. Äh, wo eigentlich? Ach ja, SAP Logon. Geschafft, und jetzt das Anmeldebild. Wie? Ach ja, SAP GUI. Aber was soll das mit diesen Mandanten? Wir sind doch keine Anwälte, wir sind Entwickler!

Anmelden zum Tauchkurs: Eintauchen in die SAP-Welt	52	Die Auslage des SAP-Systems – SAP GUI-Aufbau	61
Alles ist Transaktion – Transaktionen als Shortcuts	54	Alles auf einen Blick – SAP Easy Access	63
Melde dich an! – SAP-Anmeldung	55	Modus Operandi – Zusätzliche Fenster öffnen	65
Transaktion starten	57	Hüben und drüben, nichts als Üben – Rolands Folterkammer oder Übung macht den Meister	67
Willkommen Schrödinger! – SAP-Willkommenstext	58		

Kapitel 3: Ciao a tutti! (Hallo Welt!)

Der 20-Minuten-Einstieg in die ABAP-Programmierung

Seite 69

Endlich, das erste ABAP-Programm! Schrödinger war sich bislang gar nicht sicher, ob der Schwaiger Roland überhaupt programmieren kann. Und dann legt er auf einmal richtig los: Pakete anlegen, Datenbankzugriffe, Ausnahmebehandlung, Schlüsselwortdokumentation. Dieses ABAP fängt an, Spaß zu machen, weil es so viel zu entdecken gibt.

Einsteigen und anschnallen!	70	Kühlschrankdesign – DB-Tabelle anlegen	91
Object Navigator – Die integrierte Entwicklungsumgebung	72	Daten verarbeiten – Programm anlegen	95
Entwicklung organisieren – Systemlandschaft, Änderungsauftrag	73	Das ABAP-Einmaleins – Grundlegende Syntax	96
Ihr neuer Auftrag lautet: Auftragsanlage	75	Der Durchblicker – Programm entwickeln	98
Organisationskapsel Paket	77	Her mit den Daten – DB-Zugriff	100
Vom Entwickler zum Pakettier – Paket anlegen	79	Jetzt lese ich	101
Einen Schritt zurück und bald zwei nach vorne	82	Ohne Struktur läuft nichts – Grundlegende Programmstruktur	102
Die Infos zum Aufbau einer Datenbanktabelle ...	83	Alles muss es auch nicht sein – Einfaches Selektionsbild	104
Das zweischichtige Domänenkonzept	84	Layouten und Daten transportieren – Ein einfaches Dynpro	108
Technisches Fundament eines Typs – Domäne anlegen	86	Dekorieren der Auslage – Dynpro-Layout	110
Lege die Bedeutung an – Datenelement anlegen	89	Ablauflogik ohne Ablaufdatum – Ablauflogik programmieren	116
		Ein Shortcut für den User – Transaktionscode anlegen/ausführen	120

Kapitel 4: AAA – Alles außer ABAP

ABAP-Infrastruktur und -Werkzeuge

Seite 121

Der Schrödinger hat es schon gemerkt: Zum ABAPen reicht es nicht aus, sich die Sprache ABAP selbst reinzuziehen. Denn wo liegen die Programme, welche Programme gibt es eigentlich, womit werden sie bearbeitet? Welche Speicherbereiche werden genutzt? Wie kommen die Programme zu den Anwendern? Und ungefähr weitere 1.000 Fragen. Aber der Schwaiger Roland ist ja (noch) geduldig.

Warum?	122	Werkzeugkiste – Entwicklungswerkzeuge	137
Die Ablage der Entwicklungsobjekte – Repository	123	Object Navigator – Der Regisseur	138
Wer sucht der findet – Suchen mit dem Repository Infosystem	128	Repository Browser verschwinden lassen	139
Suche von oben – Suchen in der Anwendungshierarchie	130	Repository Browser ausnutzen	141
Suchen und Finden – Repository Infosystem	131	Repository Browser browsen	142
Geschichtet, aber nicht gefaltet – ABAP-Infrastruktur	132	Synchron oder aus dem Tritt – Objekt-Browser-Synchronisation	145
Wohin mit dem Anwender – Speicherorganisation extern und intern	135	ABAP Editor – Die Schreibmaschine	147
		Debugger – Der Kammerjäger	150
		Debugger entschleunigt	151
		Verwendungsnachweis – Der Rückwärtssucher	155

Kapitel 5: Schräge Typen*

*also jetzt nicht der Schwaiger Roland

Datentypen und -objekte

Seite 157

Definieren und deklarieren: das tägliche Geschäft des ABAP-Programmierers. Das ist ja wie mit den unterschiedlichen Nudeltypen. Die eine Sorte ist lang und dünn, andere sehen aus wie Röhren oder wie Schmetterlinge. Jedoch, und das weiß der Schrödinger natürlich, wird für ein bestimmtes Gericht oder eine besondere Pastasöße ein bestimmter Nudeltyp benötigt. Und so ist es bei der Programmierung eben auch: Für bestimmte Aufgaben werden bestimmte Typen benötigt.

Vorspann	158	Variables Standgas – Datenobjekte	159
Von 0 auf 100 in wenigen Seiten – Technik der Datenspeicherung im Hauptspeicher	158	Variablen sind zum Angreifen	160
		Ein typischer Blickwinkel – Datentypen	161

Anders betrachtet – Datentypen	163	Her mit der internen Tabelle	183
Die Vermessung	164	Wie viel Spalten hätten Sie denn gerne –	
Ansprache – Datenobjekte verwenden	165	Zeilentyp	184
Dynamik pur oder Statik nur? – Statisch und		Normal, sortiert oder doch gehasht –	
dynamisch typisierte Variablen	166	Tabellenart	185
Mein lokaler Typ	167	Open the door please – Schlüssel	187
Zuerst mal elementar lokal – Lokale		Heute mal kopflos – Kopfzeile	188
elementare Typen	168	Tabellen für alle – Globaler Tabellentyp	189
Dann elementar global – Globale		Operation: Table – Tabellenoperationen	191
elementare Typen	170	Bibliothek einräumen	193
Elementar global con domäne – Domäne	173	Einzeln geht es auch – Einzelverarbeitung von	
Strukturell lokal – Lokale strukturierte Typen	176	Tabelleneinträgen	195
Struktur global – Globale strukturierte Typen	178	Ändern und Löschen zum Abrunden –	
Tabellarisch – typisch/intern –		Noch mehr Tabellenoperationen	196
Lokaler Tabellentyp	182		

Kapitel 6: Si parla ABAP? Echtes Küchenlatein

ABAP-Syntax I

Seite 197

Jetzt geht's ans Eingemachte, Schrödinger, oder sollten wir besser sagen ans Einkochte? Hier lernst du mal das Basisvokabular, so wie damals, als nur Spaghetti gemacht wurden und nicht auch dieses andere Zeugs.

Von Kontrollen und Schleifen	198	Notandor – Logische Ausdrücke	212
Zuweisungen, Operationen, Funktionen	198	Verzweige, falls ... – IF ... THEN ... ELSE	214
Bitte nicht abweisen – Zuweisungen	201	Cassis – CASE ... WHEN ... ENDCASE	216
Was du darfst und was du sollst –		Schleifen	217
Konvertierungsregeln	203	Bedingungslose Schleifen – DO ... ENDDO	217
Ketten ohne Perlen –		Krasses Projekt für Hardcore-ABAPer	218
Zeichenketten-Funktionen	205	Bedingte Schleife – Fußgesteuert	220
Von Segmenten und Offsetten –		Bedingte Schleife – Kopfgesteuert	221
Zeichenketten-Operationen	209	Jetzt aber mal systematisch – Systemfelder	222
Ohne Rechnung, ohne mich – Arithmetische		Großbaustelle Rezeptprogramm	224
Operationen	211		

Kapitel 7: Kleine Häppchen sind leichter wiederverdaut („Schluss mit Spaghetti!“)

ABAP-Syntax II

Seite 227

Schwaiger wirkt wild und entschlossen. Er hat endgültig genug vom Spaghetti-Coding, das Schrödinger fabriziert. „Heute mache ich aus dir einen Wiederverwender“, hallt die letzte Schwaiger-Drohung durch die Hallen, und Schrödinger versucht, sich mit Wehmut von seiner letzten Spaghetti zu trennen ...

Motivation durch Demotivieren	228	Ein Typ muss es sein – Schnittstelle typisieren ...	243
Den richtigen Zeitpunkt finden – Ereignisblöcke	229	Bitte mit Typ – Typisieren	244
Ereignisreiche Programme – Ereignisschlüsselwörter	230	Warum in die Ferne schweifen? – Lokale Variablen	245
Ereignisse: Rekapitulation mit Selbstreflexion ...	231	Sichtbar oder nicht – Lokale Überdeckung	246
Zur richtigen Zeit an irgendeinem Fleck?	233	Aufruf bitte – Unterprogramm-Aufruf	247
Meine schönsten Ereignisse – Rahmenprogramm	234	Totalumbau	248
Dynpros mit Modul – PBO, PAI, HOV	236	Globale Wiederverwendung – Funktionsbausteine	252
Module, ganz kurz – Dynpro-Programmierung ...	237	Funktionsgruppe	254
Aber nun mit Schnittstelle – Unterprogramme ...	238	Leg dir eine Funktionsgruppe an	257
Definiere zuerst! Unterprogramm-Definition	239	Es ist so weit, global bereit – Funktionsbaustein	258
Übergeben und Übernehmen – Schnittstellen- parameter für das Unterprogramm	240	Her mit dem Rezept	260
Lesen von DB mit Kapsel – DB-Zugriff in Unterprogramm	242	Haaalloooooo – Funktionsbaustein-Aufruf	263
		Dein Finale	265

Kapitel 8: Schrödinger zeigt Klasse

ABAP Objects I

Seite 267

Schrödinger ist heute nicht gut aufgelegt, weil der Schwaiger Roland mit der Objektorientierung daherkommt. Es funktioniert doch alles bestens mit dem „prozeduralen ABAP“ (so nennt das der Schwaiger Roland). Was hat der bloß? Na okay, ein paar Mal haben sich schon Fehler eingeschlichen, die schwer zu finden waren. Mit der Objektorientierung soll ja alles besser werden mit der Ordnung. Und wo mehr Ordnung ist, ist alles leichter zu finden – sogar die Fehler. Und außerdem kann man mit ABAP Objects anscheinend tolle GUIs realisieren. Nach dem ersten Schock-erlebnis ist Schrödinger also gleich überzeugt.

Motivation zur OO	268	Mit Methode – ran an den Code	300
Begriffe aus der Realität	269	Die Schnittstelle der Methode	304
Holistische Sichtweise	271	Ganz spezielle Methoden	307
Modellierung	272	Methoden mit Fülle	309
Erst denken, dann lenken: Einfache UML als Analysewerkzeug zur Visualisierung von Klassen/Objekten	276	Selbstreferenz	312
Klasse Kaffee(-vollautomat) auf dem Papier	278	Statisches Attribut	313
Ohne meine Kollaborateure bin ich nichts	280	Methodenaufruf	314
Klasse oder doch Objekt	282	Laufzeit sequenziert	316
Klasse Lokal	284	Methoden für den Wasserbehälter	318
Dein kleines Objektistan	287	Von lokal nach global	320
Kaffee für zwei	290	Going global – SE24	321
Datenablage – Attribute	292	Attribute und Methoden	322
Wasserstand und Pause	294	Kaffeebohnen im Behälter	326
Klasse Attribute, oder was?	295	Senden und Empfangen	330
Privat verkalkt/Private Attribute	298	Sender mit Sendungsbedürfnis	331
Ab jetzt mit Methode – Schrödinger frischt auf ...	299	Empfänger	332
		Objektorientiertes Verkuppeln	333
		Kaffee s'il vous	335

Kapitel 9: Erben oder nicht, das ist hier die Frage: Pasta di Erbe

Vererbung

Seite 337

Hoffentlich ist der Schrödinger heute in der passenden Stimmung. Eine Vertiefung der Objektorientierung, puh, da wird er schwitzen. „Was, da geht noch mehr? Jetzt hab ich mich gerade erst vom ersten OO-Schock erholt!“ Schrödinger wird wohl erkennen, dass Spaghetti, Ravioli, Tortiglioni etc. alles Nudeln sind, oder?

Motivation	338	Redefinieren – Polymorphie von Methoden	355
Vererben und Erben von Klassen	341	Redefinieren als globale Herausforderung	358
Vererbung lokal	344	Statische und dynamische Typen von Klassen mit ein wenig Casting	364
Meister der Pyramiden	346	Festigung der Einsichten	366
Globale Vererbung: Ein praktisches Beispiel im SAP-Standard	348	Down-Cast	369
Was wird wie und wo vererbt	351	Abstrakte Klassen	372

Kapitel 10: Keine Details bitte

Der Blick durch Interfaces

Seite 377

„Man kann auch noch einen Schritt weiter gehen und die Implementierung von Methoden von deren Schnittstellendefinition trennen“, verkündet Roland und legt eine Kommunikationspause ein. Schrödinger steht da, der Mund ist offen, und er ist sich noch nicht der Tragweite der Aussage bewusst.

Schizophrenie?	378	Von Suchmaschinen und Tilden	387
Lokal zum Einsteigen	381	Hast du gerufen?	388
Schnittig und definiert	382	Die vielen Gestalten der Methoden	390
Volles Risiko – alles oder nichts	383	Globalisierung mit	
Klasse Zuordnung	384	positiven Auswirkungen	393
Klasse Definition	385	Global klasse Implementierung	395
Klasse Implementierung	386	Singleton-Party	397

Kapitel 11: Das Kapitel für Feiglinge

Ausnahmen

Seite 401

„Fehlerbehandlung ist für Feiglinge! Das ist ja so, als ob ich beim Nudelkochen die Nudeln koste, ob sie al dente sind oder nicht!“ Schrödinger hält nichts von Ausnahmebehandlungen. Bis zum ersten Zwischenfall: als seine Freundin sich über klebrige und zerkochte Nudeln beschwerte.

Ausnahmsweise behandelt	402	Ausnahmen melden (mit Nummern)	414
Eine kleine Geschichte der Ausnahmen	404	... und behandeln (nach Nummern)	415
Nochmal langsam: Mit SY-SUBRC.		Ausnahmslos Objekte	417
Ausnahmewert wird gesetzt	408	Definieren geht vor	419
... und von dir behandelt	409	Ausnahmen melden (mit Ausnahmeklassen)	421
It's RAISING man, hallelujah.	410	... und Ausnahmen behandeln, aber richtig!	
Ausnahmen melden	410	Mit Ausnahmeklassen.	424
... und behandeln	411	Das ausnahmengespickte Projekt	425
The Catcher in the System	414		

Kapitel 12: Spaghetti-Programmierung

Selektionsbilder, Listen und Nachrichten (GUI I)

Seite 431

Es ist so einfach, eine SAP-Oberfläche zu programmieren: Ein Selektionsbild erstellen, eine Liste ausgeben, vielleicht noch ein paar Nachrichten an den Anwender verschicken – und fertig ist das berühmt-berüchtigte SAP GUI! Das klingt nach Spaghetticode und ist damit ganz nach dem Geschmack von Schrödinger. Zum Einstieg in die GUI-Programmierung ist es der beste Weg, einfach mal ein paar Elemente auf dem Bildschirm auszugeben. Für das richtige Ambiente sollen dabei Klettererdbeeren und Vogerlsalat sorgen ..., sagt der Schwaiger Roland.

Vom kleinen Window-Farmer bis zum Groß-GUI-Besitzer: Jeder will ein Selektionsbild!	432	Mehr, Mehrere, Mehreres (ohne Rauschen)	453
Selektionsbild für Beschränkte mit Liste	434	Zur Anwendung gebracht ... fast	456
PARAMETERS: Das kleinere der Selektionsbild-Ungeheuer	436	Graue Theorie: Eingabehilfe, Prüftabelle	458
Typ und Vorschlag	439	Entfesse den Künstler in dir: Screen-Gestaltung	460
Hey, Checker!	441	Endlich Texte!	462
Rund wie ein Radiobutton	443	Wie im Hamsterrad	464
„Du darfst“ war gestern, heute ist „Du musst“	445	Ereignisreich geht's weiter	470
Abflug in den Keller	446	Tagesschau ... also eine Nachrichtensendung	473
Aber satt war er noch immer nicht: SELECT-OPTIONS	449	L – Li – Lis – List – Liste	477
		Keine Beichte notwendig: Interaktion mit einer Liste	484

Kapitel 13: Lasagne aufs Auge

Dynpro-Programmierung (GUI II)

Seite 487

„Wow, ein dynamisches Programm!“ Das gefällt Schrödinger sofort, dieses Dynpro. Dynamik ist einfach klasse, außer natürlich, es geht um Sport. Mit der Ablauflogik, Elementen zur Gestaltung der Darstellung und Eigenschaften, die man selbst programmieren kann: Super! Das ist schon richtig modular. (Ja, so langsam gewöhnt sich Schrödinger an die Feinheiten im ABAP-Vokabular.) Und es sind auch schon Schichten erkennbar, die Bausteine zur Dynpro-Programmierung sehen fast schon aus wie eine geschichtete MVC-Architektur. Und Schrödinger liebt Schichten, vor allem in Form von Lasagne!

Dynamisch programmieren	488	Alles ist im Fluss und manchmal geht es im Kreis	497
Wer schreit hier so? Rahmenprogramme, die Dynpros rufen!	492	Wohin soll ich mich wenden?	501
Dreieinigkeit	495	Wo Module bestimmen	502

Mit welchem Titel darf ich dich ansprechen?	505	Dein Dynpro	525
Über der Fläche steht die Oberfläche	508	Sammle Elemente	527
Die Gestaltung eines eigenen Menüs	511	Wohin mit den Daten?	529
Kannst du mit dem Druck umgehen?	514	Ablauflogik Reloaded	531
FFFFF	517	Dynpro rufen mal anders	539
Weck den Künstler in dir	519		

Kapitel 14: Ravioli

Web-Dynpro-ABAP-Programmierung (GUI III)

Seite 543

Heute wartet ein besonderer Leckerbissen auf Schrödinger: Web Dynpro ABAP. Webanwendungen erstellen ohne HTML-Kenntnisse. Das klingt gut, findet Schrödinger. Ist aber gerade nicht so wichtig, findet der Chef. Kann man gut online lesen, finden alle. Also: Schau auf der Bonus-Seite unter <http://www.sap-press.de/3024>

Bonusseite:

<http://www.sap-press.de/3024>

Motivieren funktioniert nicht	544
-------------------------------------	-----

Kapitel 15: Raus aus meiner Küche!

Berechtigungen

Seite 547

„Hör mal, Schwaiger Roland, zu viele Köche verderben doch bekanntlich die Nudelsonne. Gibt es Möglichkeiten in ABAP, dass nicht jeder alles darf? Also dass ich bestimmten Anwendern manche Aktionen verbieten oder erlauben darf? Wenigstens einschränken?“ Der Schwaiger Roland lächelt nur weise.

Berechtigungsrundumschlag – Überblick		Der Berechtigungs-Selbstchecker –	
Berechtigungen	548	S_TCODE prüfen	556
Am Anfang steht das Objekt mit Klasse –		Experimente mit AUTHORITY-CHECK	558
Berechtigungsobjekt	551	Warum mag mich keiner – SU53	559
Die Details zur Berechtigung	554		



Kapitel 16: Vorratskammer einrichten mit ziemlich viel Schnickschnack

DB-Tabellen erstellen

Seite 561

Was geschieht, wenn Anwender alle Daten im GUI eingegeben haben? Wohin gehen die Daten dann? Die müssen doch gespeichert werden? Denn die Anwender werden wahrscheinlich nicht immer wieder die gleichen Daten eingeben wollen – außer wenn sie an einem schwachen Kurzzeitgedächtnis leiden. Das würde sich Schrödinger öfter wünschen, das mit dem Kurzzeitgedächtnis, denn dann könnten sich die Anwender nicht an seine Programmierfehler erinnern ..., aber der Schwaiger Roland erklärt ihm dann doch lieber, wie er eine SAP-Datenbanktabelle erstellt.

Freiland-Datenhaltung – Daten persistieren	562	Ich will auch anders suchen – Sekundärindex	586
Warum einfach, wenn es mit Schnittstelle geht –		Ändern oder nicht, was geht –	
Die Datenbankschnittstelle	564	Erweiterungskategorie	589
Transparente Tabellen en Detail	567	Definieren und Realisieren –	
Spalten und der Rest – Tabellenfelder	570	Datenbankobjekt	591
Ohne Technik keine Tabelle – Technische		Mein erster Eintrag – Datenbanktabellen-	
Einstellungen	575	Einträge erzeugen	592
Mehr als eine Tabelle	578	Artenvielfalt im Dictionary –	
Welcher Wert ist möglich – Fremdschlüssel	581	Weitere Tabellenarten	594

Kapitel 17: Vorratskammer plündern

DB-Tabellen auslesen

Seite 595

Daten rein, Daten raus, und das möglichst einfach: So wünscht sich das der Schrödinger. Da kann ihm der Schwaiger Roland helfen. Und alleine die Begriffe „Open SQL“ und „ANSI SQL“ klingen wie Musik in seinen Ohren. Und er kann dann Daten aus einer oder sogar mehreren Datenbanktabellen lesen. Manchmal wünscht sich Schrödinger, er könnte den Schwaiger Roland in der Vorratskammer einsperren. Nur über Nacht.

Erster Takt – SQL	596	Alles recht und schön – Berechtigungen und	
Zweiter Takt – SQL	596	Konsistenzprüfungen	598
Eingelagert und geplündert – Datenpufferung	597	Open SQL grundiert – Basisbefehlssatz aus	
Datenmanipulator nativ – DML mit Native SQL	598	Open SQL	599
Einfach definieren – Datendefinition mit		Verklausuliert – SELECT-Klausel	601
dem ABAP Dictionary	598	Wohin damit – INTO target-Klausel	604

Woher – FROM source-Klausel	605	Da will ich auch nicht alles –	
Aber bitte nicht alles – WHERE-Klausel	607	Selektionsbedingungen	627
Da geht noch mehr – Weitere Klauseln	608	Zusammenfassung in Bildern –	
Und jetzt alles kombiniert – SELECT-Beispiel	609	View-Definition	628
Mehr als eine Tabelle ist auch okay – Views	614	Mit Views programmieren	630
Schritt für Schritt zum View mit relationalen		Weil du so brav warst – Programmieren	
Operatoren	618	mit Views	632

Kapitel 18: Vorratskammer in Schuss halten

DB-Daten pflegen

Seite 635

Daten pflegen – den Ausdruck mag Schrödinger. Und denkt an die Kräutersammlung auf seinem Fensterbrett. Da müssen manchmal auch neue Kräuter eingefügt werden, mal muss er das eine durch das andere austauschen. Und für das Löschen hat er auch schon eine leckere Soßenidee. Datenpflegeservice Schrödinger!

Lesen ist Silber, Schreiben ist Gold –		Und noch eins und noch eins und ... – INSERT ...	650
DELETE, UPDATE, MODIFY und INSERT	636	Massendaten einfügen	652
Sichere Daten – Transaktionskonzept	636	Ändern muss man auch ab und zu – UPDATE	655
Von einem Zustand zum nächsten –		Massendaten	656
Datenbank-LUW	638	Weg damit – DELETE	658
Bist du konsistent?	641	Kombianweisung – MODIFY	661
Aktionen bündeln – SAP-LUW	642	Halt, Sperre! – Sperrkonzept	663
Und jetzt auch noch transaktional –		Die fünf Gebote der performanten	
SAP-Transaktion	643	DB-Programmierung	664
Datenmanipulator-Entwicklungsrahmen –			
Das Programm für die Open-SQL-Anweisungen ...	645		

Kapitel 19: Mit Schirm, Charme – und vielleicht noch einem Melonensorbet

Daten in Dateien, Datenablage ohne DB

Seite 665

„Warst du schon einmal bei einem richtig guten Italiener?“ Schrödinger ahnt nichts Gutes. „Also so einem, wo du bereits am Eingang dem Kellner deine Garderobe abgeben kannst.“ – „Ja, das Prinzip der Garderobe ist mir bekannt, wieso?“ – „Na ja, wäre es nicht klasse, wenn die Anwender in ABAP auch Dateien einlesen oder herun-

terladen könnten – also an der Garderobe abgeben könnten?“ Au backe, das ist ja mal eine miese Metapher. Aber es ist vielleicht trotzdem wichtig zu wissen, wie man mit Daten umgeht, wenn man kein Datenbanksystem hat.

Daten ohne Datenbank	666	Download now!	676
GUI-Loads – Upload und Download	666	Einen hamma noch – Upload	679
Frontend Services	668	Daten auf dem Applikationsserver	681
Pfad ermitteln	671	OPEN House – OPEN DATASET	682
Download now, zumindest vorbereitet –		DatenTRANSFER – Daten schreiben	683
Download vorbereiten	673	READ DATASET – Daten lesen	685
Auch nett für Datenwiederverwendung –			
Clipboard	674		

Kapitel 20: Täglich wechselnde Speisekarten

Dynamische Programmierung

Seite 687

Morgens frische Lebensmittel auf dem Markt einkaufen, mittags daraus eine Speisekarte zaubern. Jeden Tag neu. Schrödinger bekommt Hunger, aber darum kann es dem Schwaiger Roland jetzt ja nicht gehen. „Was genau ...“ – „Wenn du erst während der Laufzeit eines Programms die Informationen erhältst, die für die Ausführung des Programms nötig sind, zum Beispiel den Namen einer Datenbanktabelle, dann ...“ – „Okay, okay, ein neues Konzept. Sag mir, wie es heißt, aber dann geht es in die Kantine.“

Dynamische Programmierung hat nichts mit		Dynamisches Feld	697
Beweglichkeit zu tun	688	Dynamischer Typ	697
Feldsymbole	688	Dynamische Komponente	698
Datenrefs	690	Dynamische Bedingung	698
RTTS = RTTI + RTTC	692	Dynamisches Unterprogramm	700
Überschrift finden	693	Programmerzeugung im Hauptspeicher	700
Dynamische Tokens	695	Programmerzeugung im Repository	701

Bildnachweis **704**

Index **705**

Vorwort

Der Schrödinger ist bei mir eingezogen und lebt nun seit geraumer Zeit – seit Anfang 2011 – als achttes Familienmitglied bei uns. Nicht nur im Haus, sondern auch in unseren Köpfen. Die tägliche Frage: „Was wird Schrödinger heute wieder anstellen?“ beschäftigt uns schon beim Frühstück.

Dabei hat alles so harmlos angefangen. Stefan Proksch, mein geschätzter Lektor der Web-Dynpro-Bücher, fragte mich, ob ich denn nicht Lust auf eine Bekanntschaft mit Schrödinger hätte. Ein umgängliches Kerlchen, mit Ecken und Kanten, der bereits die C++-Schule bei Dieter Bär besuchen durfte. Das konnte ich mir natürlich nicht entgehen lassen. Naiv und enthusiastisch, wie ich bin.

Am Anfang war der Sturm und Drang, so wie in jeder frischen Beziehung, und der Schrödinger und ich hatten eine intensive Phase des „Aneinander Gewöhnens“.

Ja, ja, da waren wir noch jung. Mittlerweile ist daraus eine solide Beziehung entstanden, die darauf ruht, sich aufeinander verlassen zu können. Dazu hat sicher auch unsere Lektorin Almut Poll beigetragen. Sie hat unsere Beziehung gestärkt, geglättet, gehegt und gepflegt.

Aber es geht ja nicht um mich, sondern um dich – oh, du geschätzter Wegbegleiter. Schrödinger und ich werden dich mit in die ABAP-Fabrik nehmen, und du bekommst eine ausführliche Werkstour. Apropos Werkstour, diese beginnt mit einem Hinweisschild am Empfang. Der Text darauf lautet so:



Was darfst du nicht erwarten: eine verstaubte Betriebsführung. Also, wenn du bereits auf die „üblichen“ Betriebsführungen konditioniert bist, dann darfst du leider nicht eintreten.

Keine Chance. Der Sicherheitsdienst wird dich sofort enttarnen. Also **bitte umdrehen, sofort.**

Du bist noch da. Das freut uns aber sehr. Du hast die erste Hürde elegant genommen.

Was darfst du also erwarten: eine Einführung in die essenziellen Handgriffe im SAP-System, einen ABAP-Schnelldurchlauf, mit dem sogar ein Formel-1-Pilot überfordert wäre, dann alles außer ABAP und alles mit ABAP. Also Typen, Variablen, Syntax, Abzweigungen, äh, Verzeihung, Verzweigungen, (Schuld-)Zuweisungen nach Operationen, gepflegte Ausdrücke und Lichterketten. Routinierte Ereignisse vor klasse Objekten. Einige Bilder für den User, und das sogar im Web. Datenbank und Tabellen, schreiben und lesen, und nicht die Dynamik zu vergessen, die ABAP wandlungsfähig machen. Dazu noch einiges an Betriebssport und -küche und ein gelegentlicher Besuch im Kreativzentrum.

Viel Spaß!

Ups, da hätte ich ja fast etwas vergessen. Mein Dankeschön, und das betrifft dieses Mal nicht nur die üblichen Verdächtigen. Und wie immer weiß ich nicht, wie ich anfangen soll ... vielleicht eine kleine Geschichte zuerst: Stefan Proksch, mein Web-Dynpro-Lektor und Schreiblehrer, hat ihn mir persönlich vorgestellt und war sehr geheimnisvoll dabei. Offiziell durfte ich ihn – den Schrödinger – gar nicht kennen und musste sogar unter Folter seine Existenz leugnen. Das hab ich dem Stefan sogar unterschreiben müssen. Aber leugnen und foltern hin oder her, der Schrödinger hat sich bei mir immer mehr eingenistet. Und damit das so richtig nachhaltig war, hat Stefan mir auch unermüdlich immer wieder neue Facetten vom Schrödinger vorgestellt. Gott sei Dank war er geduldig genug mit mir. Der Schrödinger war mir am Anfang nicht geheuer. Und dann hat Stefan auch noch unsere Männer-WG verlassen, und da schauten der Schrödinger und ich mal kurz etwas dumm.

Doch welch Glück: Innerhalb kürzester Zeit ist ein neues WG-Mitglied eingezogen: die Almut Poll. Und da waren wir wieder drei: die Almut, der Schrö und der Roland. Von da an ging's aber hurtig dahin, und die Almut hat den Schrödinger und mich so richtig nach vorne gebracht. Danke Almut, dass du dem ganzen Projekt nochmal richtig Schwung gegeben hast. Deine Einfälle und Anmerkungen haben mir so manches Mal aus der Patsche geholfen. Danke!

Es gibt noch mehr, die mich begleitet haben:

Dr. Gerhard Rodé möchte ich für die informative Mail-Kommunikation zu der Entwicklung von ABAP danken. Einiges ist ins Buch eingeflossen, und im Download-Bereich wird dazu noch mehr zu finden sein.

Meinem geschätzten Referentenkollegen Dr. Stefan Ehret gebührt Dank für seine hoch qualitativen Schulungsunterlagen und die Erlaubnis zur Verwendung seiner dynamischen SE16.

Martin Schwaiger, mein Bruderherz, hat mir schönen Input für die Kaffeegeschichten geliefert. Danke dafür, und hoffentlich wird unser **Espresso Stout** ein Wintertraum.

Meinem Sohn Nico für die Schrödinger-Zeichnungen und Schrödinger-Ideen („Papa, was macht der Schrödinger heute wieder ...?“) und dafür, dass er Teile des Buches geschrieben hat ;-)

Meiner Tochter Marie für den Schrödinger-Gugelhupf, der wie immer auch dem Schrödinger schmeckte (der hat ihn alleine verputzt).

Meiner Tochter Elisa für die schicken Schrödinger-Kekse.

Dir, Ursula, weiß ich nicht, wie ich danken soll, denn zu viel wäre es, was ich schreiben dürfte. Danke für meine Freiräume und für den ganzen Rest. Und ich verspreche, dieses Jahr kein Buch mehr zu schreiben. ☺

Widmen möchte ich dieses Büchlein meinem Vater, von dem ich lernte, durchzuhalten.

—DREI—

Der 20-Minuten-
Einstieg in
die ABAP-
Programmierung

Ciao a tutti!
(Hallo Welt!)

Endlich, das erste ABAP-Programm!
Schrödinger war sich bislang gar nicht sicher, ob der
Schwaiger Roland überhaupt programmieren kann.
Und dann legt er auf einmal richtig los:
Pakete anlegen, Datenbankzugriffe, Ausnahmebehandlung,
Schlüsselwortdokumentation. Dieses ABAP fängt an,
Spaß zu machen, weil es so viel zu entdecken gibt.

Einsteigen und anschnallen!



Es ist wahrscheinlich ratsam, dass du dir eine Tasse **Kaffee** (Betonung auf dem letzten e) oder ein **Erfrischungsgetränk** deiner Wahl zubereitest und bereitstellst.

Zur Auswahl steht alles mit Durchhaltefaktor > ein Tag:

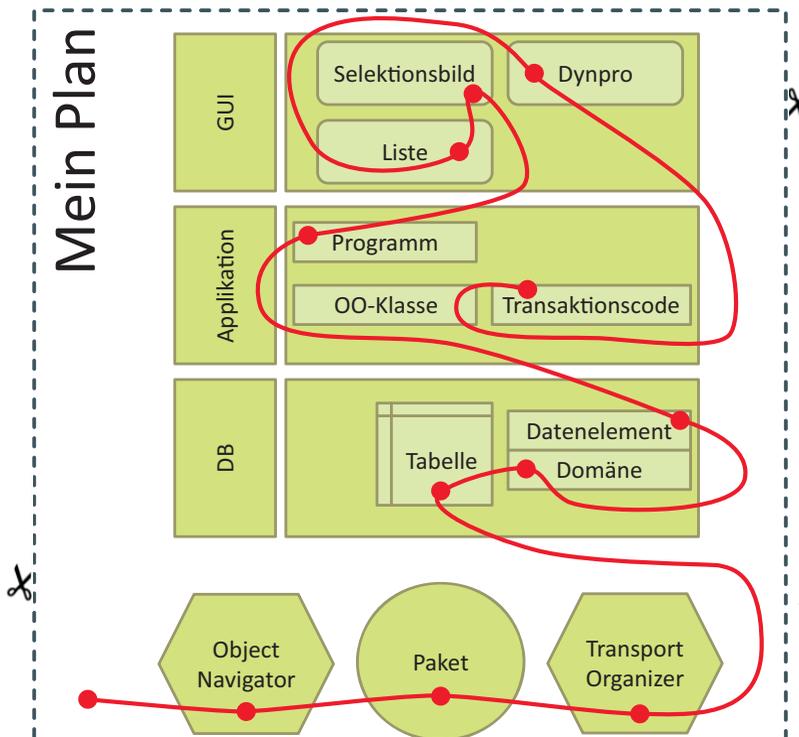
- ☛ Kaffee, gemessen nicht in Tassen, sondern Kannen
- ☛ schwarzer Tee, direkt bezogen von der eigenen Teeplantage
- ☛ Energydrink, vorrätig in Paletten

Dieses Kapitel benötigt deine volle Konzentration und Energie. FOKUS!

Ich werde dich mit **unglaublicher Geschwindigkeit** durch das **ABAP-Universum** ohne Pause („We break for nobody!“) navigieren, und wir werden alle Planeten des Inner Rims besuchen, wobei ich das Steuer unseres Raumschiffs übernehme.

Bereit für den Start? Noch schnell einen **Schluck**, und los geht's. Wie ich sehe, bist du ja bereits am SAP-System angemeldet.

Mein Plan für dieses Kapitel ist eigentlich nicht umsetzbar. Aber durch einen genialen Trick habe ich die Verständlichkeit extrem erhöht: Ich habe einen **didaktischen roten Faden** eingefädelt!



Cooler Plan, oder?
Ich habe dir Markierungen zum Ausschneiden eingefügt. Schneide doch einfach den Plan aus, damit du nicht immer hin und her blättern musst.

Scherzkeks!

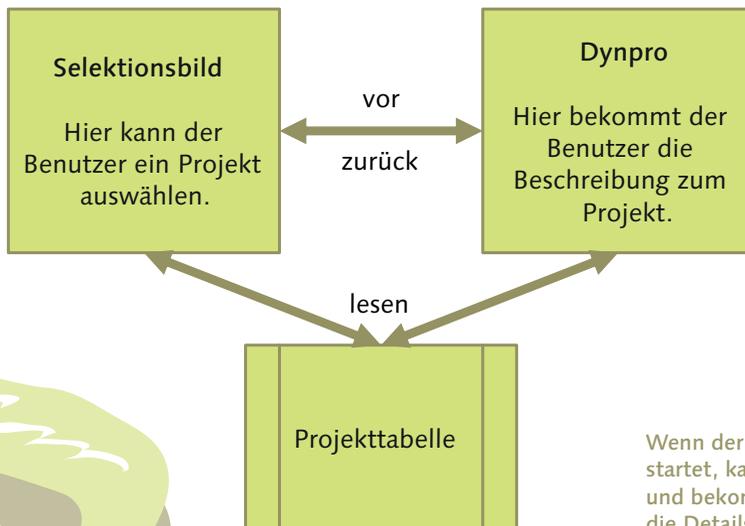


Na gut, dann einige Details dazu:
Um dir einen Einblick und Durchblick zu verschaffen, möchte ich dir gerne beschreiben, wie du so ein Kunstwerk schaffen und in Form eines Projektes umsetzen kannst.

Sehr beeindruckend für Fensterbrett, Balkon und Garten. Durchblick und Einblick in einem.



Die Beschreibung werde ich natürlich im SAP-System in einer **Datenbanktabelle** hinterlegen und dir auf einem **Dynpro** anzeigen. Da das Programm und die Datenablage sehr allgemein ausfallen werden, kannst du sie auch für deine eigenen Projekte als Beschreibungsbasis verwenden.



Wenn der Anwender das Programm startet, kann er ein Projekt auswählen und bekommt nach der Bestätigung die Details zum Projekt angezeigt.

Der Anwender kann in einem **Selektionsbild** ein Projekt auswählen und Details dazu anzeigen lassen. Dafür wird ein Dynpro verwendet. Die Daten kommen natürlich, wie es sich gehört, aus einer Datenbanktabelle.

Falls dein Motor schon warmgelaufen ist, legen wir einen Le-Mans-Start hin, sprinten von der Couch zum Schreibtisch und werfen die Transaktion zum Programmieren an.

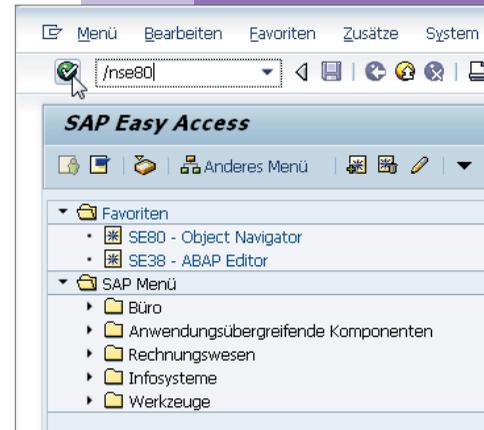
Object Navigator – Die integrierte Entwicklungsumgebung

Das Werkzeug, von dem aus du alle deine **Entwicklungen** starten wirst, ist der **Object Navigator**, der die Transaktion **SE80** besitzt [Rührt sich da was im Kopf?]. Damit hast du auf alle weiteren Transaktionen Zugriff, die für die **Bearbeitung** von **Entwicklungsobjekten** benötigt werden. Der Object Navigator ist deine **Schaltzentrale**: Anlegen, Ändern, Löschen, Umbenennen, Aktivieren, Kopieren, und vieles mehr wird dir zur Verfügung gestellt.

1. Tippe im Transaktionscode-Feld **/nse80** ein, und bestätige deine Eingabe.

Starten der Motoren mit der SE80.

2. Schau dich etwas um in der SE80.
Für uns sind derzeit nur einige Teile wichtig.
 - Der **Navigationsbereich** dient dir zum Auswählen von Entwicklungsobjekten.
 - Die **Objektliste** zeigt dir die ausgewählten Entwicklungsobjekte an.

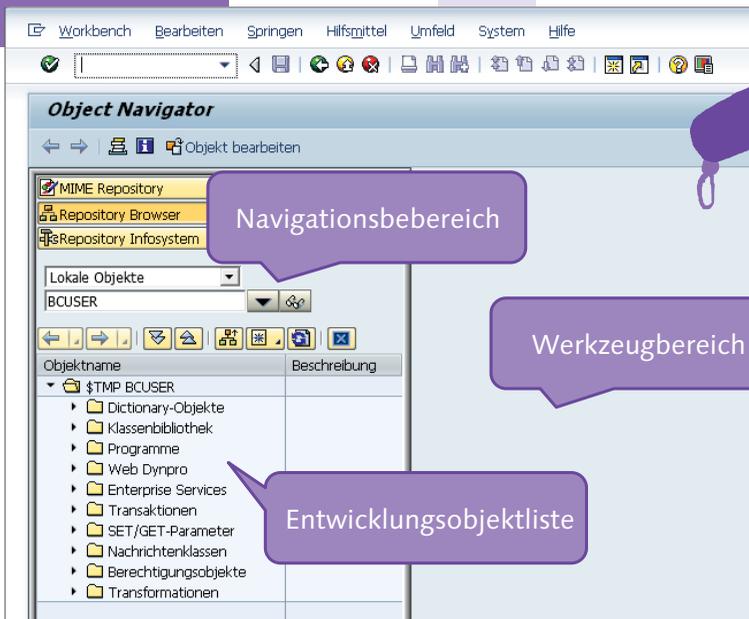


- Der **Werkzeuggestrich** dient zum ...? Wozu wohl?

[Einfache Aufgabe]

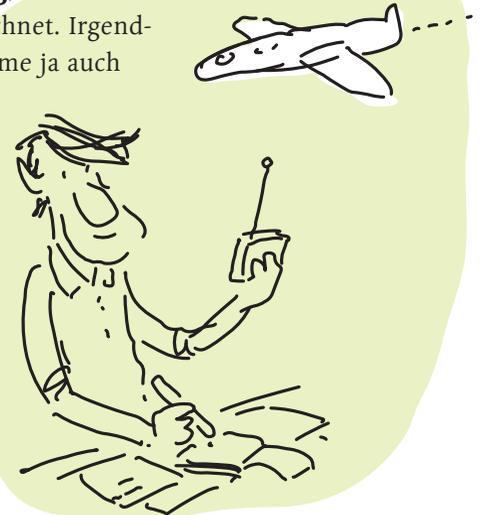
Sieh dich in der SE80 um, mach dich mit den Bereichen vertraut, versuche hier und dort einen Doppelklick. Falls du dich verläufst, rufe einfach wieder **/nse80** auf. Das ist wie bei Hänsel und Gretel mit den Brotkrümeln, wobei du nur einen hast, den du dafür aber auch sicher wieder findest.

Im Navigationsbereich kannst du Objekte auswählen, in der Objektliste dann das Ergebnis der Auswahl bewundern und im Werkzeugbereich bearbeiten.



Entwicklung organisieren – Systemlandschaft, Änderungsauftrag

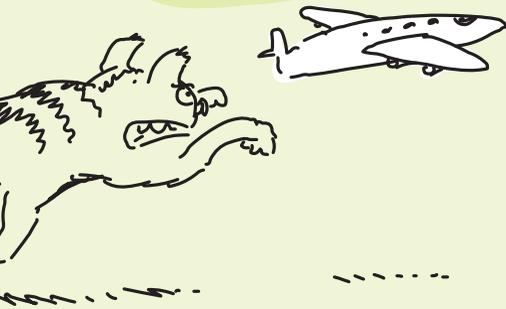
SAP empfiehlt dir und somit seinen Kunden eine **Drei-System-Landschaft**. Ein System für die **Entwicklung** – in dem werden wir uns bewegen –, ein System für die **Qualitätssicherung**, und eines für die **Produktion**, auch als **Produktivsystem** bezeichnet. Irgendwo müssen die Anwender deiner formidablen Programme ja auch arbeiten können.



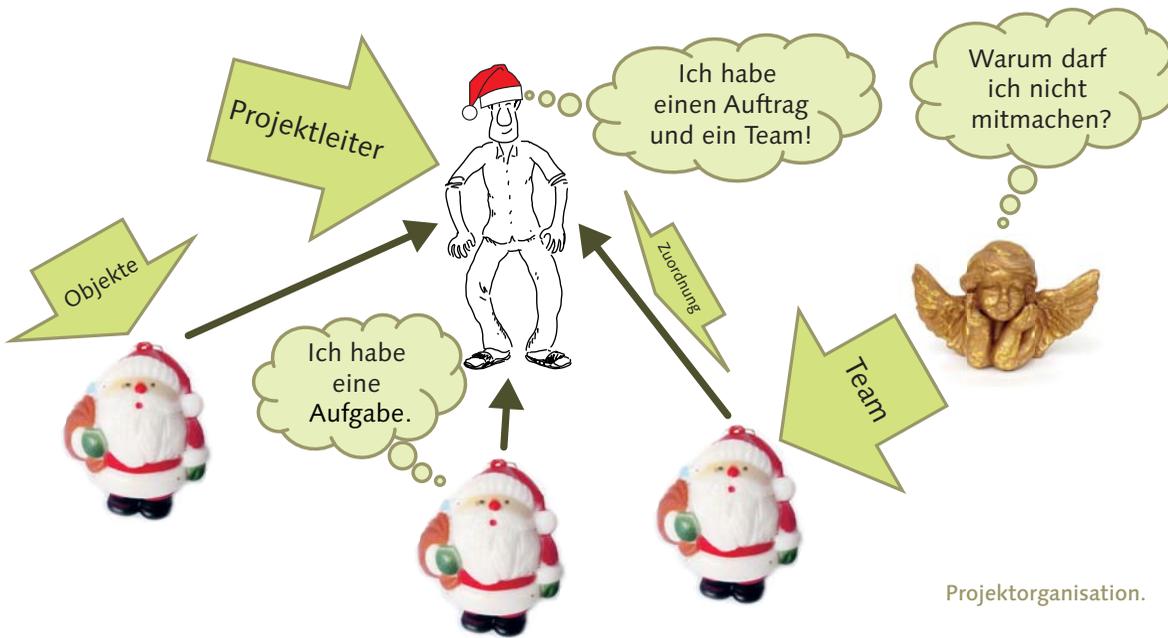
Bauen –
Testen –
Verwenden.



*Oh Schreck, da wird jemand
damit arbeiten? Schluck.*

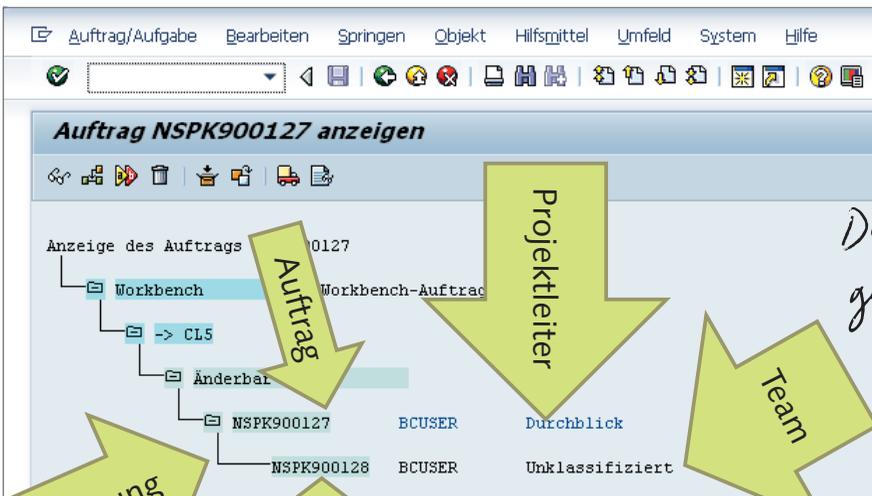


Die Entwicklungsobjekte werden im Rahmen eines **Entwicklungsprojektes** im **Entwicklungssystem** angelegt. Am Ende der Entwicklung müssen die Objekte in das Folgesystem transportiert werden. Zu diesem Zweck legt der **Projektleiter** mit der Transaktion **SE09** einen Änderungsauftrag an und ordnet die Mitarbeiter dem Auftrag zu. Dadurch wird **pro Mitarbeiter** eine **Aufgabe** im **Auftrag** angelegt. Falls du nun ein Entwicklungsobjekt änderst oder anlegst, wird dieses Objekt der Aufgabe zugeordnet. Während der Abarbeitung der Aufgaben im Projekt werden alle Objekte im Auftrag gesammelt. Mitarbeiter, die nicht Teil des Projektes sind, können die Entwicklungsobjekte leider nicht ändern.



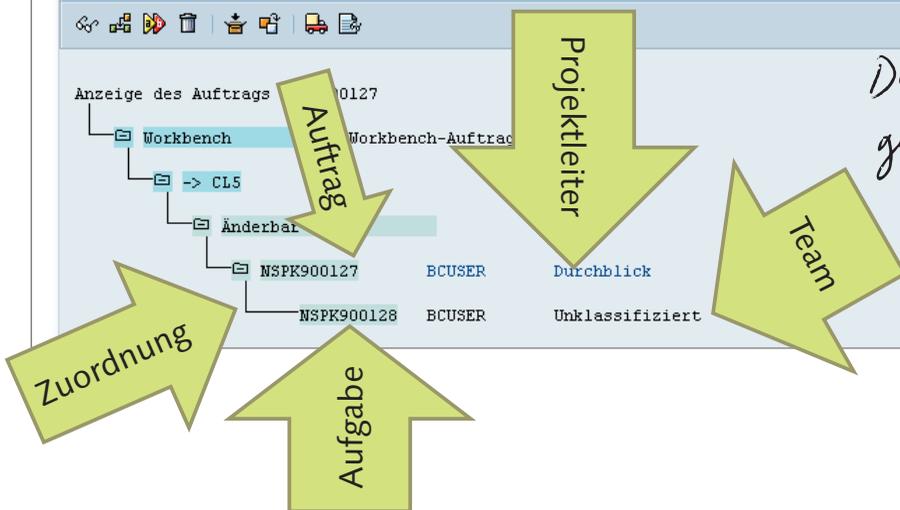
Projektorganisation.

Falls dir das mit dem Engerl zu kitschig ist, kann ich dir auch eine eher formale, nüchterne Darstellung eines Änderungsauftrags (auch bekannt als **Workbench-Auftrag**) anbieten.

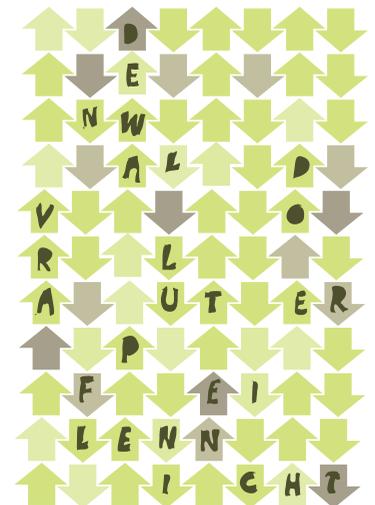


So sieht das im SAP-System aus. Gleiche Information, nur ohne W-Mann und Engerl. So einen Auftrag mit Aufgabe wirst du gleich anlegen.

Des W-Mann gefällt mir besser.



Ist das Projekt umgesetzt, kann jeder Entwickler seine **Aufgabe freigeben**. Dies führt aber noch nicht zur kompletten Freigabe des Auftrags. Die muss vom **Projektleiter** (dem Besitzer des Auftrags) erteilt werden. In welche Systeme transportiert wird, legt die **Transportschicht** fest, die dem Paket zugeordnet ist.

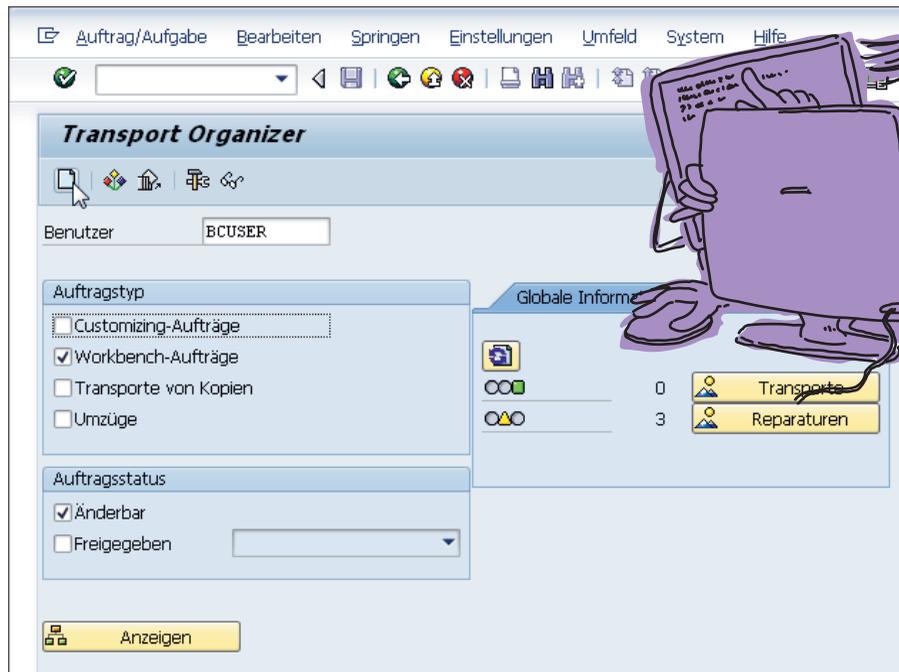


Ihr neuer Auftrag lautet: Auftragsanlage

Jetzt legen wir uns einen Änderungsauftrag für unser Projekt **Durchblick** an.

1. Ruf die Transaktion SE09 auf, am besten mit /ose09. Was geschieht beim /o nochmal?

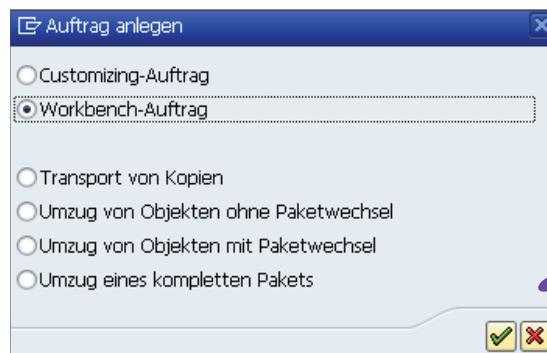
Natürlich öffnet sich ein neues Modus. Zumindest solange noch Fenster geöffnet werden können.



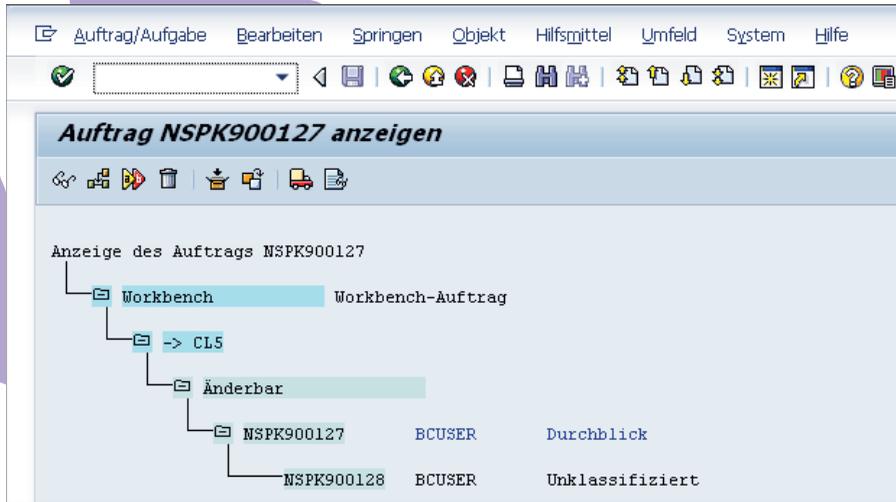
Dein erster Auftrag, den du als Projektleiter anlegst.
Die Anlegen-Drucktaste (F6) macht den Job.

2. Im folgenden Pop-up wählst du **Workbench-Auftrag**, da wir Entwicklungsobjekte anlegen werden. Bestätige dies mit dem grünen Häkchen.

Der Workbench-Auftrag ist dein Sammel-Container für alle Entwicklungsobjekte zu einem Projekt.



3. Wir kommen schon zum Ende. Im erscheinenden Pop-up gibst du noch die Bezeichnung des Auftrags ein – ich finde „Durchblick“ sehr passend –, und, da du schon automatisch als Bearbeiter zugeordnet bist, bestätigst das mit der **Sichern**-Drucktaste (Enter).



Dein Auftrag und deine Mission: Durchblick zu schaffen. Du bist Projektleiter und Mitarbeiter zugleich. Kannst du mit dieser Schizophrenie umgehen? Sei nicht zu hart zu dir, falls du den Termin nicht halten kannst.

Damit bist du für die Entwicklung gerüstet.

Auf in Windeseile zum nächsten Schritt. Das **Paket!**

*Das ist ja wie Weihnachten.
Ich bekomme ein Paket vom
SAP-Christkind!!*



Organisationskapsel Paket

Ein **Entwicklungsobjekt** kann einem **Auftrag** zugeordnet werden. Das währt aber nur kurze Zeit, nämlich so lange, bis der **Auftrag freigegeben** wird. Dann kommt schon der nächste Auftrag ... Man könnte auch sagen, dass das **Objekt einem Auftrag temporär zugeordnet** ist.

Sind wir noch im Plan?

Beim Kästchen Paket, ist das richtig?

Ja, alles noch im Plan. Es gibt noch eine andere Art der Zuordnung, nämlich eine **permanente Zuordnung zu einem Paket**. Wenn du Java kennst, wird dir der Begriff **Paket** bereits etwas sagen. Falls du Java nicht kennst, macht das auch nichts, und du darfst weiterlesen.



[Begriffsdefinition]

Ein **Paket ist ein Container**, in dem unter anderem Entwicklungsobjekte gesammelt werden. Dabei können Pakete **Schnittstellen** und **Verwendungserklärungen** besitzen. Darüber hinaus können **Pakete geschachtelt** werden.

Für die unterschiedlichen Granularitätsebenen gibt es unterschiedliche **Arten von Paketen**.

- **Standardpakete** („kein Hauptpaket“) nehmen die Entwicklungsobjekte auf.
- Standardpakete können in **Hauptpakete** geschachtelt und somit gruppiert werden.
- Hauptpakete wiederum können **Strukturpaketen** zugeordnet werden.
- **Strukturpakete** werden bei der Strukturierung großer Softwareprojekte eingesetzt. Die sind für dich als Einsteiger derzeit nicht wichtig. Ich würde vorschlagen, wir reden in einem Jahr wieder über das Thema.

Früher hat es andere Container gegeben, die als **Entwicklungsklassen** bezeichnet werden. Diese sind mit den Standardpaketen vergleichbar.

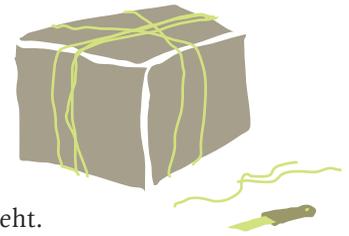


[Hintergrundinfo]

Im SAP-System gibt es reservierte **Namensräume** für Kundenobjekte und SAP-Objekte. Wenn du den Kundennamensbereich verwendest (die Namen der Entwicklungsobjekte beginnen mit Z oder Y bzw. mit einem bei SAP reservierten Namensraum, zum Beispiel /SCHROED/), werden deine Objekte während des Einspiels neuer SAP-Entwicklungsobjekte nicht überschrieben.

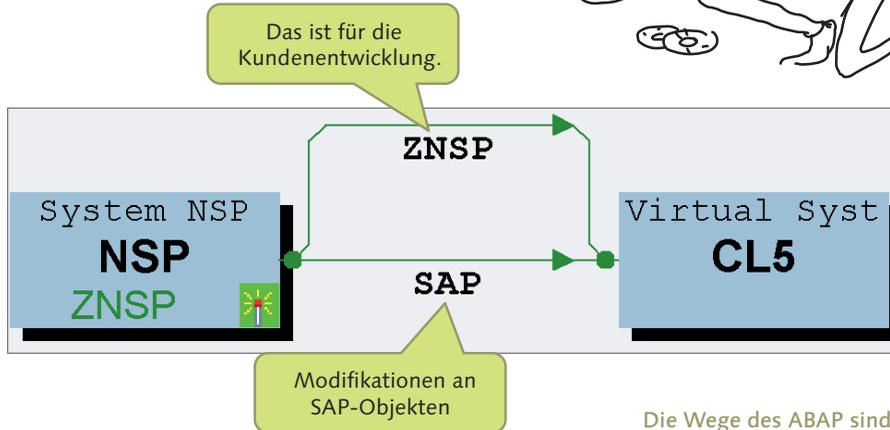
Safety first!

Die EIGENSCHAFTEN, die zu einem Paket festgelegt werden, sind:



- **Paketname**, natürlich im Kundennamensraum!
- **Kurzbeschreibung**: Hilft zu beschreiben, um was es thematisch in dem Paket geht.
- **Anwendungskomponente**: Dient der Einordnung des Pakets in die Gesamtmenge der Anwendungen – die auch als Anwendungshierarchie bezeichnet wird –, kann jedoch initial, das heißt leer bleiben.
- **Softwarekomponente**: Die Softwarekomponente beschreibt eine Menge von Paketen, die nur gemeinsam auslieferbar sind. Für Kundenprojekte wird **immer** die Softwarekomponente **HOME** verwendet.
- **Transportschicht** (ah, da ist sie): Für Kundenentwicklungen wird eine Transportschicht eingerichtet, die darüber entscheidet, ob und wenn ja, in welches Folgesystem die Objekte transportiert werden. Hier hilft dir eine Zeichnung von oben. Welche wohl? Nochmal zurückblättern?

Lösung: Die mit Basen - Testen - Versenden natürlich.



Die Wege des ABAP sind wunderbar.



[Hintergrundinfo]

Kleiner Tipp am Rande: Die Transportwege und -schichten kannst du dir mit der Transaktion **STMS** ansehen.



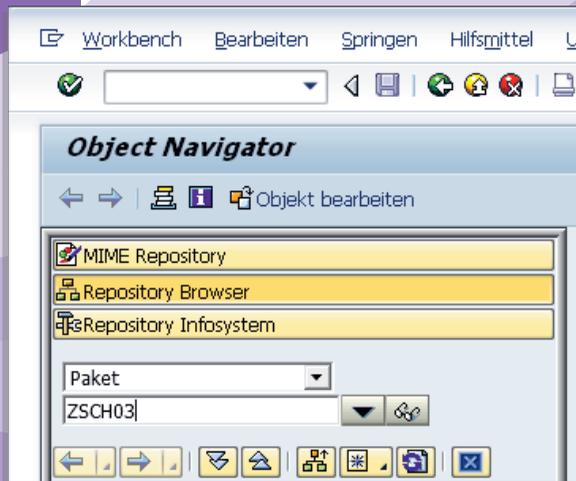
- **Pakettyp**: Es stehen drei unterschiedliche Pakettypen zur Verfügung: **Standardpaket**, **Hauptpaket** und **Strukturpaket**. Wie die Namen schon andeuten, sind die Typen für unterschiedliche Einsatzgebiete gedacht. Wir werden **Standardpakete** für die Entwicklung verwenden.

Vom Entwickler zum Pakettier – Paket anlegen

Also legen wir ein Paket an.

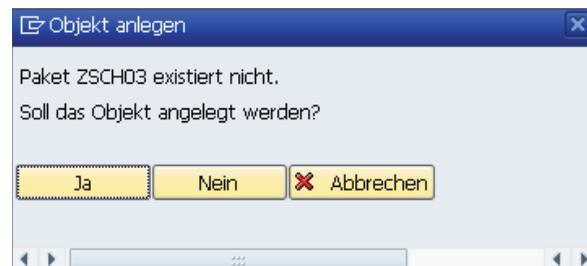
1. Wechsle in den Object Navigator. Das sollte ziemlich einfach sein, da der Modus mit dem Object Navigator immer noch vorhanden sein sollte. Wenn nicht: auch egal, einfach nochmal **SE80** aufrufen.
2. Im Navigationsbereich wählst du als Typ **Paket**, und ich schlage dir den Namen **ZSCH03** für dein Paket vor. Das ist aber nur ein Vorschlag. Was ist der einzige Aspekt, den du bei der Vergabe des Namens beachten musst?

An der ersten Stelle muss ich ein Z oder Y verwenden oder mir einen Namensraum bei der SAP reservieren. Dem wünsche ich mir zu meinem Geburtstag.



Du willst also ein Paket anlegen. Bravo, und auch noch so ein schöner Name.

3. Nachdem du die Eingaben erledigt hast, hämmerst du einfach auf die **Enter**-Taste. Und das schlaue System stellt fest, dass es das Paket noch nicht gibt. Bestätige die Neuanlage mit der Drucktaste **Ja**.



Ja, ich will anlegen, bitte.

**DRAUF
DRÜCKE!**

4. Dann kommt der Kern des Paket-Anlegeprozesses: die Pflege der Details zum Paket. Die Daten sollten aufgrund der obigen Erläuterungen klar sein. Nach den Eingaben wählst du das **Sichern**-Druckknöpfli (Enter).

Paket	ZSCH03
Kurzbeschreibung	Schrödinger mit Ein- und Durchblick
Anwendungs Komponente	
Softwarekomponente	HOME
Transportschicht	ZNSP
Pakettyp	Kein Hauptpaket

Ein neues Standardpaket (kein Hauptpaket), das keiner Anwendungs Komponente und der Softwarekomponente HOME zugeordnet wird.

5. Und jetzt kommt ein Schritt, den du dir gleich für später merken kannst. Wenn du ein Entwicklungsobjekt neu anlegst oder das erste Mal änderst, wirst du immer gefragt, welchem Änderungsauftrag, also welchem Entwicklungsprojekt, dieses Objekt zugeordnet werden soll. Und dein Vorgehen wird auch immer gleich sein. Mit der Drucktaste **Meine Aufträge** rufst du die Liste derjenigen Aufträge auf, denen du als Mitarbeiter zugeordnet bist, wählst einen per Doppelklick aus und bestätigst deine Auswahl. Als besondere Serviceleistung des SAP-Systems wird dir von nun an immer der von dir ausgewählte Änderungsauftrag vorgeschlagen.

Paket	ZSCH03	
Auftrag	NSPK900127	Workbench-Auftrag
Kurzbeschreibung	Durchblick	

Zuerst kommt die Abfrage, welchem Änderungsauftrag du das Paket zuordnen möchtest. Mit der Drucktaste **Eigene Aufträge** erhältst du eine Liste mit möglichen Aufträgen. Du suchst dir einen (Durchblick natürlich) per Doppelklick aus und bestätigst dann deine Auswahl mit dem grünen Häkchen. **Mit drei Klicks zum Ziel!**

Objekte dieses Typs können nur in einen transportierbaren Workbench-Auftrag mit Ziel "CL5" aufgenommen werden

Workbench-Aufträge mit Beteiligung von BCUSER (ddic DDIC)

- > CL5
 - Änderbar
 - NSPK900122 BCUSER NETWORK
 - NSPK900127 BCUSER Durchblick
 - NSPK900118 BCUSER THEMES
 - NSPK900124 BCUSER Schrödinger



[Erledigt!]

Du bist am Ziel angekommen. In der Objektliste im Object Navigator wird dein neues Paket angezeigt. Du hast die erste Hürde bravourös gemeistert!



Das Ergebnis von ein paar Klicks.
Ein neues Paket.

Und gleich weiter, keine Pause. Wir beginnen, uns von unten nach oben hochzuarbeiten. Vom Tellerwäscher zum Millionär oder von der Datenbank zum GUI.

Paanuse, biittle.

Okay, setz dich mal gaaaaanz locker hin.



Einen Schritt zurück und bald zwei nach vorne

Einmal losschnallen, Bleistift holen, entspannen, aber nicht einschlafen. Überlege dir Folgendes: Können Weihnachtsmann und Christkind im selben Team spielen?

Wenn du dafür eine Antwort gefunden hast, sollten die weiteren Fragen sehr einfach für dich zu lösen sein.

Die Transaktion SE09

- dient als Folterinstrument für abtrünnige ABAP-Entwickler.
- ermöglicht dir, die Entwicklung zu organisieren.
- hilft dir bei der Anlage eines Programms.
- erzeugt Zuordnungen von Mitarbeitern zu Projekten.

[X', 'X', 'X']
[Lösung]

Die unterschiedlichen Pakettypen sind:

- Vor-, Mitte-, Hinterpaket
- Eins-, Zwei-, Dreipaket
- Wichtig-, Mittelwichtig-, Vollwichtigpaket
- Standard-, Haupt- Strukturpaket
- Fein- Mittel-, Grobgranularpaket

[Vorletztes]
[Lösung]

Und jetzt wieder anschnallen,
die Reise geht weiter.

Die Infos zum Aufbau einer Datenbanktabelle

Zur Erinnerung, nachdem wir nun die nötige Infrastruktur für die Entwicklung unseres Programms haben: **Wir bauen ein Programm zur Anzeige von Beschreibungen und ein Bild zu einem ausgewählten Projekt.** Die Daten möchte ich natürlich in einer Datenbanktabelle speichern. Ich bin bescheiden, eine reicht mir. Welche Informationen sollten dort abgelegt sein? Sicher mal das Projekt, um das es geht, dann die Beschreibung und vielleicht noch Bilder dazu. Wie bekommen wir das in eine Tabelle?

Naja, das ist wie mit der Frage, wie bekomme ich einen Elefanten in einen Kühlschrank? Kühlschranktüre auf, Elefant rein, Kühlschranktüre zu.



Wunderbar, du hast das Prinzip des Arbeitens mit einer **Datenbanktabelle** verstanden. Ich möchte mit dir aber den Kühlschrank bauen.



Also ganz kurz:

Eine **Datenbanktabelle** dient zum persistenten Speichern von Daten. Die Tabellen, die ich hier bespreche, sind Teil einer relationalen Datenbank.

Eine Datenbanktabelle besteht aus Spalten, die namentlich benannt werden und einen Typ besitzen. Die Typen sind vor- oder von dir im **ABAP Dictionary** definiert, wo auch die Beschreibungen der Tabellen (sogenannte transparente Tabellen) angelegt werden. Manche der Spalten werden als Schlüssel definiert, um einen Eintrag der Tabelle eindeutig identifizieren zu können.

Zu einer Datenbanktabelle gehören gewisse technische Einstellungen, wie die Auslieferungsklasse, Pufferung etc.

Hast du die Beschreibung fertig angelegt, kannst du diese aktivieren, was zur Erzeugung der Datenbanktabelle im Datenbanksystem führt.

Du sollst dir merken: Eine Datenbanktabelle besteht aus **Spalten** und **Zeilen**.

Die Spalten sind die Daten und die Zeilen die Datensätze.

		Spalten			
		MANDT	PROJEKT	BESCHREIBUNG	BILD
Zeilen		001	001	Das Projekt Durchblick durchbohrt einen Baumstamm ...	baum.png
	

Eine Datenbanktabelle hat Zeilen und Spalten. Manche nennen diese Anordnung auch zweidimensional.

Für jede Spalte musst du beschreiben, welche Informationen dort gespeichert werden können. Ist das eine Zahl, oder ist das ein Text etc.? Du musst die Spalte typisieren und damit festlegen, welche Inhalte abgespeichert werden können.

Ich zeige dir sofort die Variante für die erwachsenen Entwickler, ohne in die Tiefe einzusteigen, weil die sowieso in Kapitel 5 kommt:

Das zweischichtige Domänenkonzept

Unser Plan: Zuerst sogenannte **Datenelemente** und **Domänen** und dann die Datenbanktabelle anlegen. Unter einer Domäne stellst du dir bitte einen technischen Typ vor, und unter einem Datenelement die Texte zu dem technischen Typ. **Alle, die jetzt schreien, bitte Mund halten!** Die Erklärung wird später verfeinert und ins rechte Licht gerückt.

Voll interessant Roland. Ich muss noch schnell eine Pause einlegen, bin gleich zurück.

Warte! Ein Wort noch: Das Zusammenspiel zwischen dem Datenelement und der Domäne wird, wie bereits erwähnt, als **zweischichtiges Domänenkonzept** bezeichnet.

[Hintergrundinfo]

Bei der Spalten-Typisierung in transparenten Tabellen oder Komponenten-Typisierungen in Strukturen werden Typen benötigt. Eine Art von Typ ist das sogenannte **Datenelement**. Das Datenelement stellt die semantische Ebene des zweischichtigen Domänenmodells dar, in dem unter anderem der Bezug zu einem technischen Typ definiert wird.



[Hintergrundinfo]

Zu der Domäne werden der **technische Typ** und mögliche Zusatzangaben zum Typ festgelegt, wie zum Beispiel **Länge**, **Nachkommastellen** etc. Natürlich noch 1.000 andere Aspekte, die jedoch erst später in Kapitel 5 verraten werden.

Du solltest dir merken: Das **Datenelement** dient zur Abbildung der **semantischen Information** und die **Domäne** für die **technische Information**.
Genug gelabert, weiter im ICE-Tempo.

Sicher nicht. Jetzt ist endgültig Pause.

Okay, dann wirf ein paar Bälle.

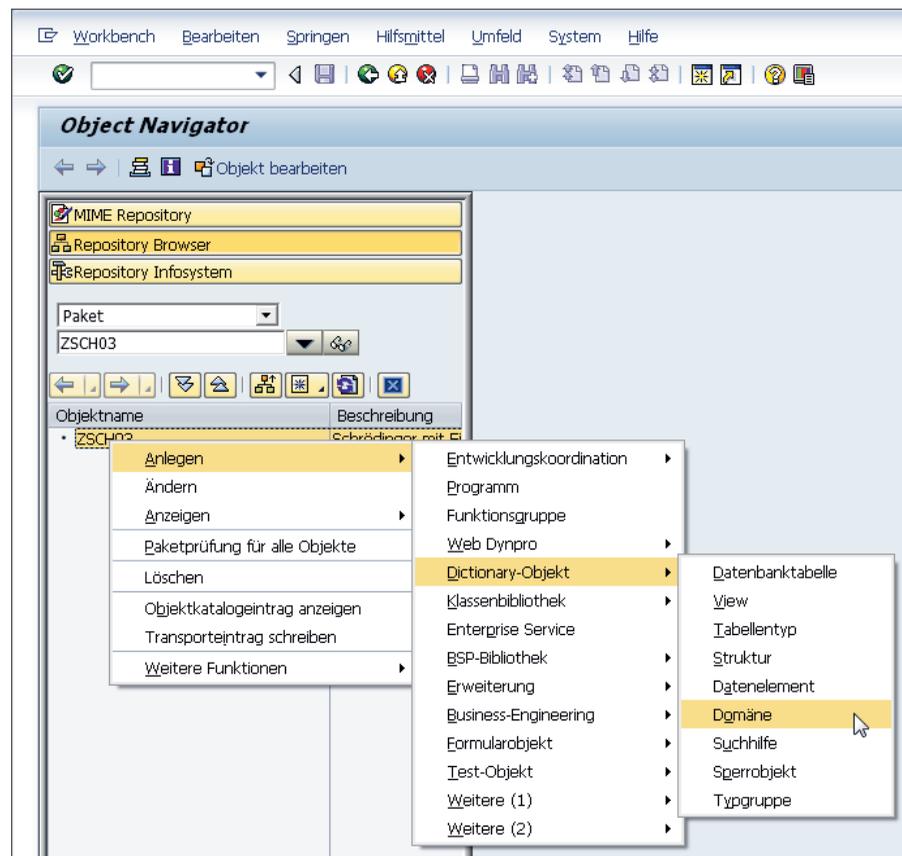
Ich bereite in der Zwischenzeit deine nächste Gehirnahrung vor.



Technisches Fundament eines Typs – Domäne anlegen

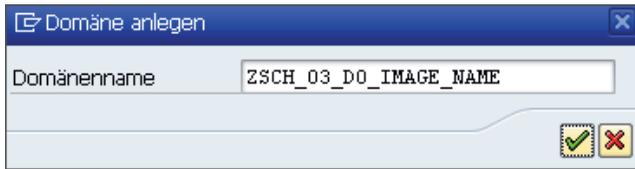
Wir werden uns jetzt eine Domäne für die Tabellenspalte zum **Bildnamen** anlegen.

1. Du startest von deinem Paket aus und verwendest den Kontextmenüpfad **Anlegen • Dictionary-Objekt • Domäne**. Damit rufst du die Transaktion SE11 im Werkzeugbereich auf. Das ist das Werkzeug **ABAP Dictionary**. Wie der Name andeutet, werden dort globale Typen wie in einem Wörterbuch hinterlegt.



Das Anlegen einer Domäne mithilfe des Kontextmenüs des Pakets.

2. Im erscheinenden Pop-up gibst du den Namen der Domäne an, **ZSCH_03_DO_IMAGE_NAME**, und bestätigst ihn mit dem grünen Häkchen.

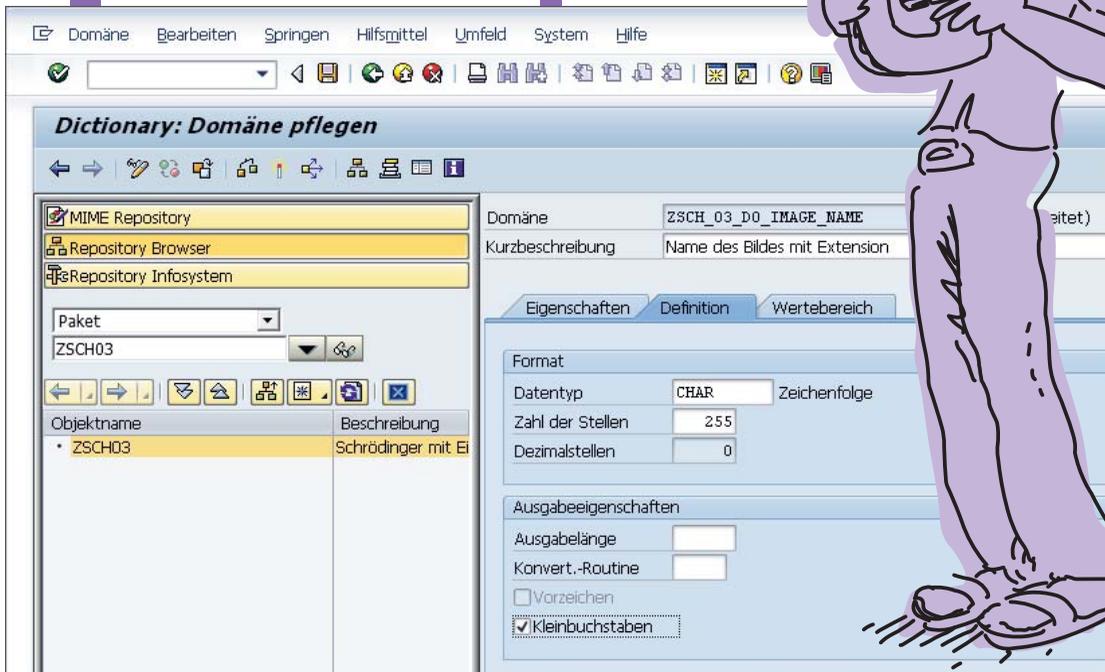
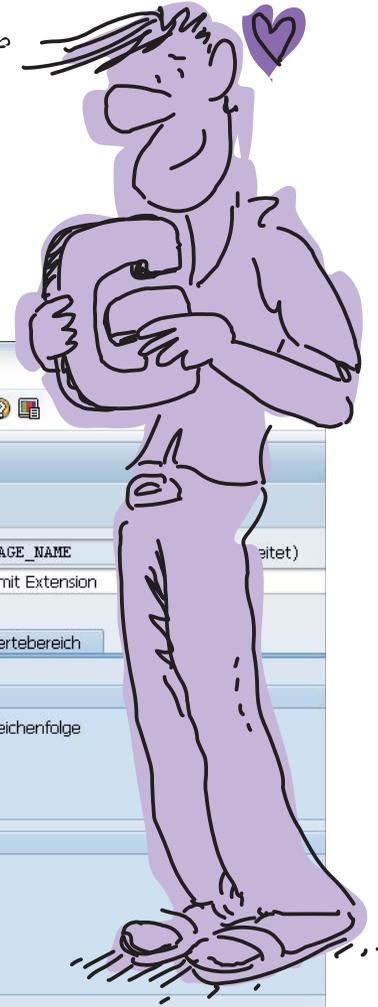


Der Name der Domäne für den Bildnamen.

3. Dann kommen die Details: Der Name sollte eine Zeichenfolge sein, mit maximal 255 Zeichen, und Groß- und Kleinschreibung sollte unterschieden werden. Das sind schon die Einstellungen, die wir benötigen, also ziemlich wenig dafür, dass jeder Entwickler im SAP-System diese Domäne verwenden kann.

Aha, das war mit globalem Typ gemeint. Global verwendbar im SAP-System. Cool.

Für die Domäne pflegst du die Kurzbeschreibung, den Datentyp, die Zahl der Stellen und die Kleinbuchstabenrelevanz.



Achtung, jetzt kommt eine Sequenz von Aktionen, die dir noch ins Blut übergehen wird: Sichern mit der **Sichern**-Drucktaste (Strg + S) in der Symbolleiste und Aktivieren mit der **Aktivieren**-Drucktaste (Strg + F3) in der Drucktastenleiste. Los, mach es!

Beim Sichern erscheint ein Dialog mit dem Titel **Objektkatalogeintrag anlegen**, und da wird das Paket ZSCH03 vorgeschlagen. Ist das okay, dass ich den Sichern-Button angeklickt habe?

Gut, dass du fragst, nachdem du etwas getan hast. Möglicherweise sollten wir noch etwas an der Frage-Aktion-Sequenz arbeiten. Du hast aber goldrichtig gehandelt. Damit wird die Domäne dem Paket zugeordnet.

Na super, dann war es sicher auch okay, dass ich im nächsten Pop-up **Abfrage transportierbares Workbench-Auftrag** mit dem grünen Häkchen bestätigt habe. Puh, Glück gehabt.

Du solltest das mit dem ICE-Tempo möglicherweise nicht zu wörtlich nehmen. Aber du hast Recht, die Zuordnung zum Durchblick-Auftrag war auch okay.

Das Aktivieren hast du sicher auch schon gemacht, wenn nicht, dann **AKTIVIEREN**.



Rechts neben dem Namen der Domäne sollte jetzt der Zustand **aktiv** aufleuchten.

[Erledigt!]

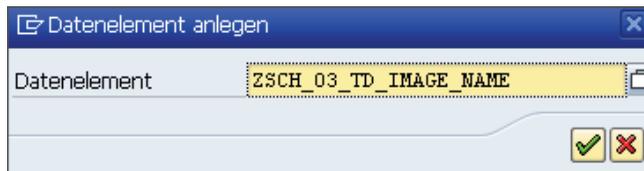
So schnell kann es gehen. Du hast deine erste Domäne angelegt. Perfekt. Als Nächstes kommt die Verwendung in einem Datenelement. SAP sagt dazu auch die ...ntische Ebene.

Romantische? Scherzes!
Natürlich semantische!

Lege die Bedeutung an – Datenelement anlegen

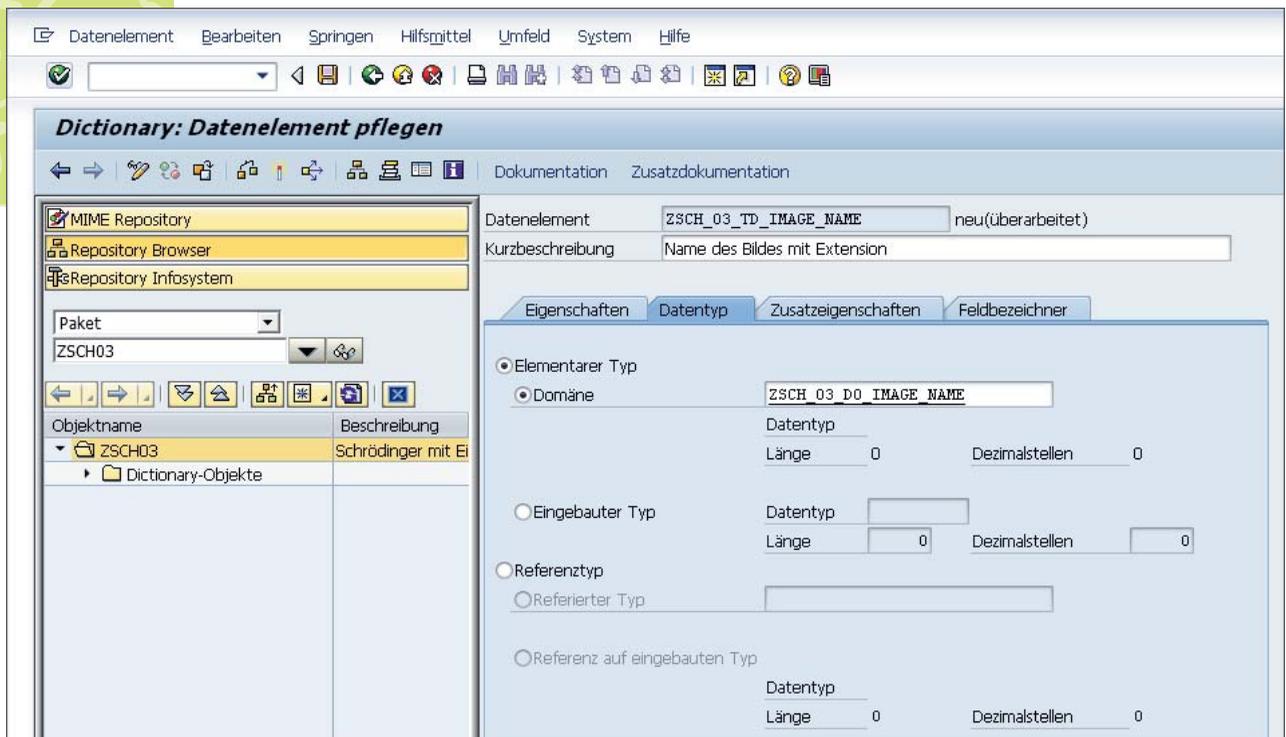
Wir benötigen ein Datenelement für die Typisierung der Tabellenspalte **Bildnamen**. Also produzieren wir es:

1. Starte wieder mit dem Kontextmenü des Pakets, und wähle den Menüeintrag **Anlegen • Dictionary-Objekt • Datenelement anlegen**. Vergib wie bei der Domäne den Namen für dein Datenelement, nur dieses Mal **ZSCH_03_TD_FILE_NAME**.



Das Datenelement. Worin besteht der Namensunterschied zwischen Domäne und Datenelement?

2. Im Folgebild pflegst du auf dem Karteireiter **Datentyp** die Kurzbeschreibung und das Eingabefeld **Domäne**. Dreimal darfst du raten, was da reinkommt? **NA; NA; NA?** Ja, genau, der Name der Domäne: **ZSCH_03_DO_IMAGE_NAME**.



Hier wird der Bezug vom Datenelement zur Domäne hergestellt. Eben zweischichtiges Domänenkonzept.

3. Wechsle auf den Karteireiter **Feldbezeichner**. Dort kannst du noch **Bezeichner** mit unterschiedlichen Längen hinterlegen, die vom System verwendet werden, zum Beispiel auf Dynpros (das sind GUIs).

Noch schnell die Texte in unterschiedlichen Längen zum Datenelement gepflegt, damit wir diese später automatisch im GUI nutzen können.



4. Wenn du die Texte fertig gepflegt hast, sind wir wieder so weit. Sichern und aktivieren, so wie vorhin bei der Domäne. Ich sagte bereits, dass das deine Lieblingssequenz wird.

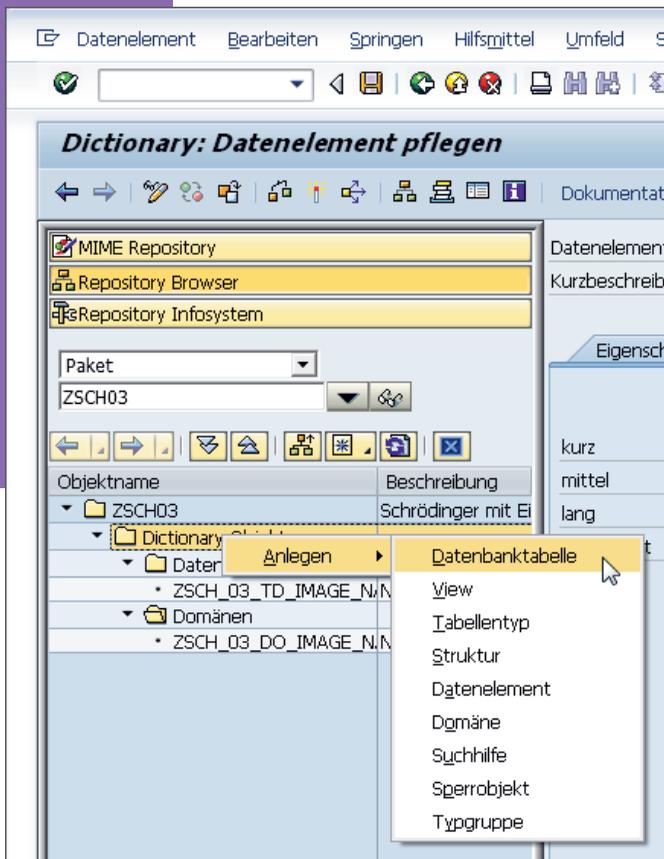
Kühlschrankdesign – DB-Tabelle anlegen

Alles ist bereit und vorbereitet. Lasst uns froho u hund munter die Tabelle **ZSCH03PROJECT** anlegen. Mittlerweile muss ich dir den ersten Schritt nicht mehr ansagen, oder?

Kontextmenü des Pakets?

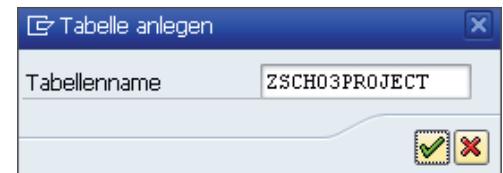
Genau!

1. Alternativ könntest du das Kontextmenü des Unterordners **Dictionary-Objekte** verwenden, um die Datenbanktabelle anzulegen. (Der Eintrag gefällt mir nicht. Müsste eigentlich **Beschreibung der Datenbanktabelle** lauten.)



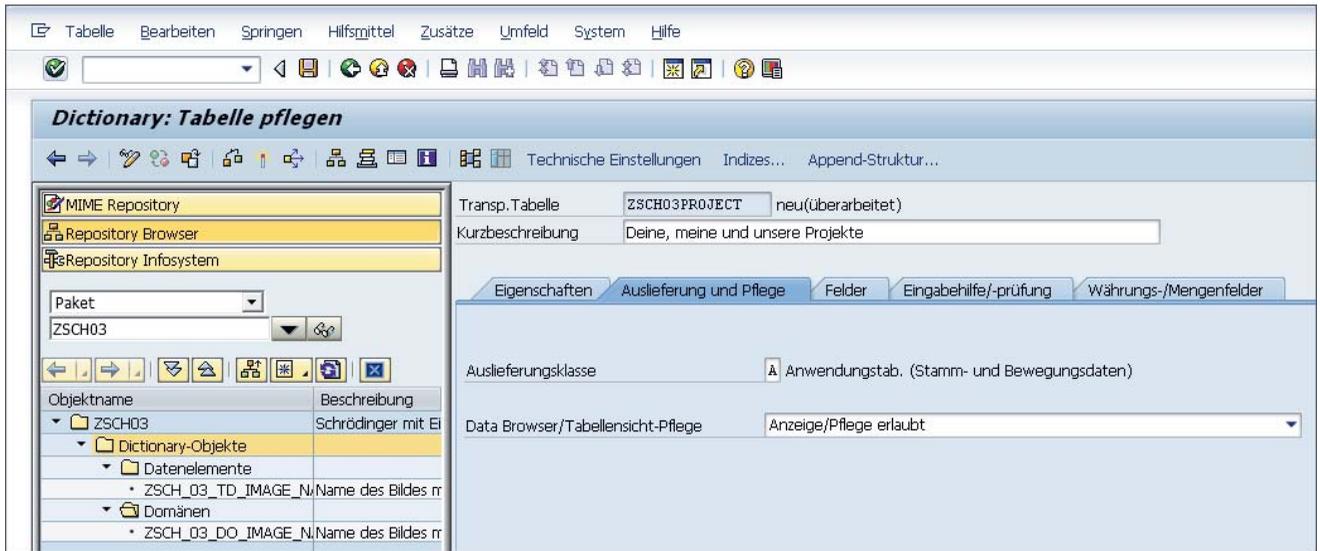
Ein alternativer Pfad zur Anlage eines Dictionary-Objektes. Jetzt geht es um Datenbanktabellen.

2. Im folgenden Pop-up vergibst du den Namen der Datenbanktabelle **ZSCH03PROJECT** und bestätigst deine Eingabe.



Der Name der Datenbanktabelle, in der die Projekte persistiert werden.

3. Auf dem Karteireiter **Auslieferung und Pflege** setzt du das Eingabefeld **Auslieferungsklasse** auf den Wert „A“, da wir kontinuierlich Daten einpflegen wollen, und **Data Browser/Tabellensicht-Pflege** auf „Anzeige/Pflege erlaubt“, damit wir Werte auch von Hand eingeben können.



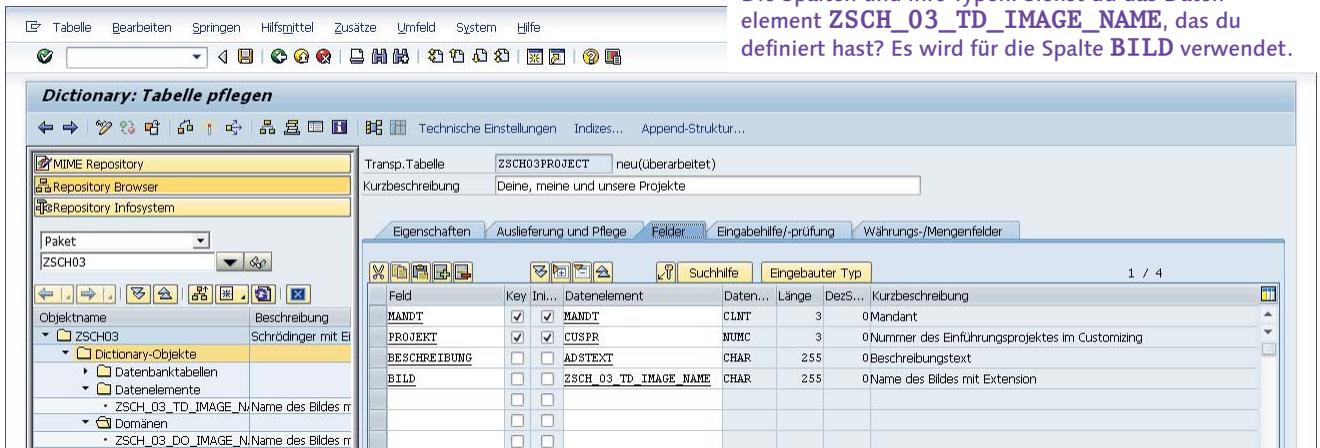
Welche Art von Daten und manueller Pflege, das sind die Fragen!

4. Wechsle auf den Karteireiter **Felder**, dort kannst du die Spalten definieren. Diese werden untereinander aufgelistet. Ich weiß, das ist komisch, du kannst sie dir ja gegen den Uhrzeigersinn um 90° gedreht vorstellen, dann sieht es wie Spalten aus.

Also bitte. Ich stelle nicht mal die Landkarte auf den Kopf, wenn ich nach Süden laufe.

Beim Pflegen gibst du den **Namen des Feldes** ein – das ist der Spaltenname –, und dann in der Spalte **Datenelement** eben das Datenelement für den Typ der Spalte. Falls das Feld ein Teil des **Primärschlüssels** ist, um Einträge eindeutig zu finden, musst du das Häkchen für **Key** setzen. Ich habe die zwei Spalten **MANDT** und **PROJEKT** als Primärschlüssel definiert.

Die Spalten und ihre Typen. Siehst du das Datenelement **ZSCH_03_TD_IMAGE_NAME**, das du definiert hast? Es wird für die Spalte **BILD** verwendet.



Dictionary: Technische Einstellungen pflegen

Überarbeitet <-> Aktiv

Name: ZSCH03PROJECT (Transparente Tabelle)

Kurzbeschreibung: Deine, meine und unsere Projekte

Letzte Änderung: 05.09.2010

Status: neu (nicht gesichert)

Logische Speicher-Parameter

Datenart: APPL0 (Stammdaten, transparente Tabellen)

Größenkategorie: 0 (Erwartete Datensätze: 0 bis 580)

Pufferung

Pufferung nicht erlaubt

Pufferung erlaubt, aber ausgeschaltet

Pufferung eingeschaltet

Pufferungsart

Einzelsätze gepuffert

generischer Bereich gepuffert (Anzahl Schlüsselfelder:)

vollständig gepuffert

Datenänderungen protokollieren

Schreibzugriff nur mit Java

5. Jetzt folgen noch einige administrative Einstellungen. Beginnen wir mit dem Menüeintrag **Springen • Technische Einstellungen**. Dort wird unter anderem festgelegt, wo die Daten in der Datenbank physisch zu speichern und wie viele Einträge zu erwarten sind. Trage bitte im Eingabefeld **Datenart** den Wert **APPL0** und in der **Größenkategorie** den Wert **0** ein.

Wo werden die Daten gespeichert, und wie viele Einträge werden es wohl werden? Das sind deine Leitfragen in den technischen Einstellungen.

6. Noch schnell **Sichern** (Strg + S), und dann wieder zurück mit dem **grünen Pfeil** (F3) in die Pflegesicht der Tabelle.

7. Eine Einstellung ist noch vorzunehmen, die **Erweiterungskategorie**. Diese legt die zukünftig erlaubten Änderungsmöglichkeiten fest. Wähle dazu den Menüeintrag **Zusätze • Erweiterungskategorie**. Das unnütze Zwischen-Pop-up kannst du mit dem grünen Häkchen ignorieren und setzt im nächsten Pop-up den Wert auf „beliebig erweiterbar“.



Jaja, alles erweiterbar. Danke und übernehmen.

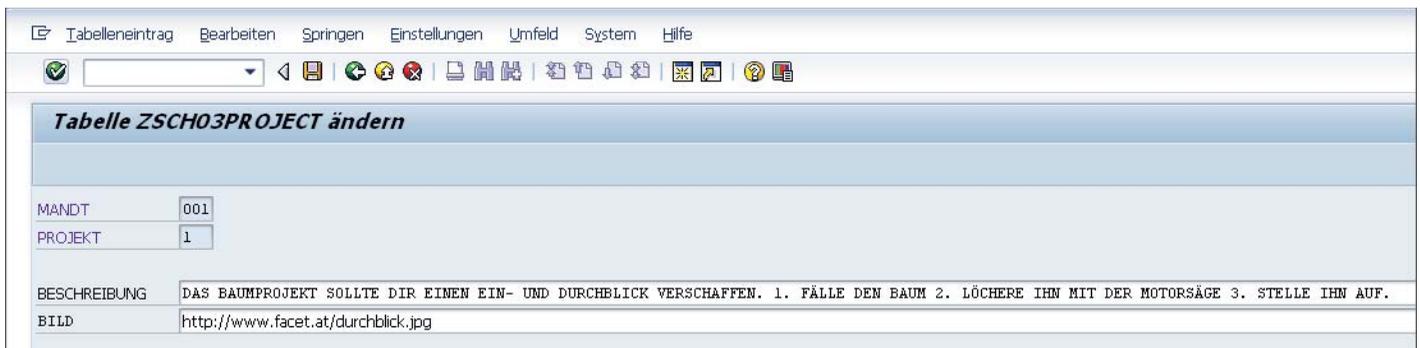
8. Nachdem du die **Übernehmen**-Taste gedrückt hast kommt wieder deine Lieblingssequenz: Sichern und aktivieren. Fertig!

Bravo, du hast deine erste Datenbanktabelle angelegt. Möchtest du gerne einen Eintrag anlegen? Okay, ganz einfach.

9. Wähle den Menüeintrag **Hilfsmittel • Tabelleninhalt • Einträge erfassen**.

10. Erfasse deine gewünschten Werte, und sichere (Strg + S) deine Eingaben.

Fertig ist der erste Eintrag.



Ein Eintrag in deiner Tabelle. Das Lochbaumprojekt.

Also das mit den Daten haben wir mal erledigt. Damit haben wir die DB-Schicht in meinem Plan „Mein Plan“ abgeschlossen. Als Nächstes wechseln wir auf die Applikationsebene und sehen uns dort ein paar Sachen an.

[Belohnung]

Vorher musst du dich aber noch stärken. Entweder bei einer Schweinshaxe mit einem kleinen isotonischen Malzgetränk oder bei einer Pilates-Übung auf der Matte.

Daten verarbeiten – Programm anlegen

Ich habe dir ja schon erklärt, wie man ein Entwicklungsobjekt anlegt; beim **Programm** Anlegen kommen noch einige Spezialitäten dazu. Du bist schon an der richtigen Stelle in der **SE80**, um zu starten.

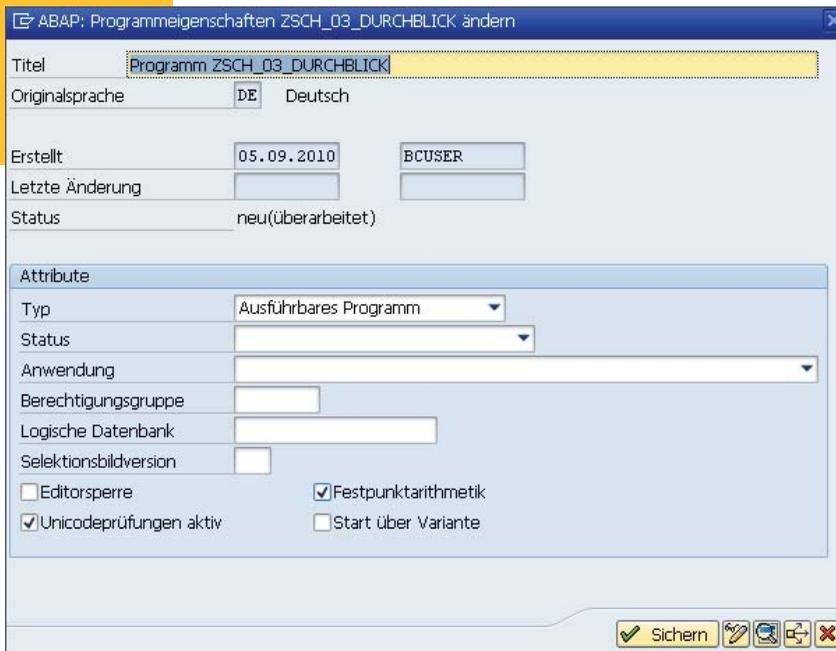
Los geht es mit dem Kontextmenü auf dem Paket **ZSCH03** „und du wählst dort den Menüpunkt **Anlegen • Programm**.



Ein Programm in seiner reinsten Form – ohne alles. 

Du wirst vom System im nächsten Schritt gefragt, ob du ein Programm mit oder ohne sogenanntem **TOP-Include** anlegen möchtest. Bevor du das entscheidest, solltest du vielleicht erfahren, was ein **Include** ist.

Das heißt, ein Include ist eine **Textkonserve**. Nicht mehr und nicht weniger. Im Include stehen Programmtexte, die beim Übersetzen des Programms eingebunden werden. Das TOP-Include ist ein Spezialfall eines Includes. Dort sollten die **Typ Definitionen** und **Daten Deklarationen** angelegt werden. TOP deswegen, weil diese Defs/Deks am Anfang/Kopf eines Programms gebündelt stehen sollten. Du willst **kein TOP-Include anlegen**, also **deselektiere die Checkbox!** Bestätige mit dem grünen Häkchen.



Nach deiner Bestätigung erscheint ein Dialog mit den Eigenschaften, den wir derzeit getrost mit der Drucktaste **Sichern** (Enter) bestätigen können, und den darauffolgenden Dialogen zur Zuordnung zum Paket und Änderungsauftrag können wir ebenfalls unser Vertrauen schenken und auf das **Sichern**-Symbol klicken.

Endlich erscheint der **Programmkopf** auf der rechten Seite im **Werkzeugbereich**. Die Transaktion, die geöffnet wurde, heißt **ABAP Editor**, die du auch mit dem Transaktionscode **SE38** öffnen kannst.

Was kann man denn da alles einstellen? Wichtig für jetzt ist nur, dass du ein **Ausführbares Programm** anlegst.

You are ready to rumble!

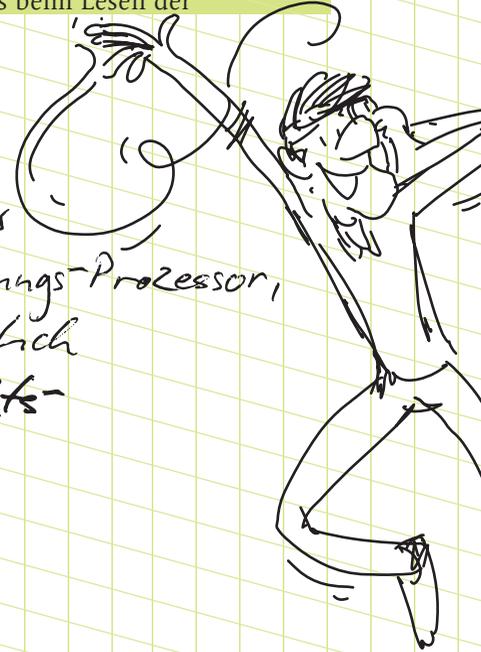
Das ABAP-Einmaleins – Grundlegende Syntax

Ein Schnellstart in die Syntax, möglicherweise eine Wiederholung für dich?

Um ABAP zu verstehen, solltest du nicht vergessen, was ABAP ursprünglich bedeutete:

Allgemeiner-Berichts-Aufbereitungs-Prozessor! Lass es beim Lesen der folgenden Zeilen immer mitschwingen.

Schawing Prozessor, Schawing
großes grünes Aufbereitungs-
Prozessor, Schawing großes grünes
Brommen des Berichts-Aufbereitungs-Prozessor,
Schaschawing großes grünes Leszlich
Brommen des Allgemeines-Berichts-
Aufbereitungs-Prozessor!



- ABAP ist typisiert.
- ABAP stellt SQL zur Verfügung.
- ABAP ist objektorientiert.
- ABAP ist aufwärtskompatibel.
- ABAP is Written Once and Truly Run Everywhere; man könnte auch sagen: ABAP ist WORA (Write Once Run Anywhere).



Der **mächtigste Befehl** in ABAP ist – **Schawing** – das **WRITE**-Statement. Die Entwicklung von mehrsprachigen Anwendungen ist ein Klacks, und die Zugriffe auf die Datenbank sind easy-cheesy. Die prozeduralen Betonköpfe müssen sich sputen, da die OO-Gazellen ihnen davon galoppieren.

Ein weiterer beeindruckender Punkt ist die **Aufwärtskompatibilität**, die sich unter anderem darin äußert, dass Programmteile aus R/2 immer noch im aktuellsten Release laufen. **Nix deprecated!**

Die Syntax sieht mal prinzipiell so aus:

Schlüsselwort **Zusatz** **.** **(Punkt)**

Der **Punkt** kommt immer am Ende einer **Programmzeile**, nicht unbedingt Editorzeile, aber sicher Programmzeile. Da werden dir am Anfang ganz komische und eigenartige Meldungen vom Syntaxchecker begegnen, falls du den Punkt vergisst. Also falls das passiert, einfach durch visuelle Inspektion (**Poorman's Syntax Check**) prüfen, ob immer ein Punkt gesetzt wurde. Punktlandung, punktgenau auf den Punkt gebracht.

Einige Beispiele zum Flexibilisieren der Synapsen:

```
DATA gd_you_can_do_it TYPE abap_bool.
```

Die Deklaration einer Variablen mit dem Namen **gd_you_can_do_it** und mit dem Typ **abap_bool**

```
PARAMETERS pa_car TYPE s_carr_id.
```

Ah, **PARAMETERS**, noch ein **deklaratives Schlüsselwort** (übrigens **immer im Plural**), dann die Bezeichnung eines Eingabefeldes mit dem Typ **s_carr_id**

```
gd_you_can_do_it = 'X'.
```

Der Variablen wird ein Wert in Form eines **Textliterals** zugewiesen.

Sicher ist dir schon aufgefallen, dass die Teile der Anweisung durch zumindest ein **Leerzeichen** voneinander **getrennt** sind. Auch in diesem Fall ist der Syntaxchecker unbarmherzig und liefert ebenso schräge Meldungen.

Komplett egal ist die Groß- und Kleinschreibung, = iRrElEvAnT, man könnte auch sagen, dass ABAP **nicht case sensitive** ist.

```
WRITE 'Warum nicht öfter etwas Nettes schreiben?'.
```

Da ist die **Königin der ABAP-Anweisungen**: Das **WRITE**-Statement befüllt den **Listpuffer** und stellt dem Anwender etwas in der **Liste** dar. Das **Etwas** ist das zwischen den **einfachen Anführungszeichen (Literalbegrenzern)**.

Du sollst deinen Sourcecode **kommentieren**, das sage ich immer zu meinen Kindern. Ja, nur WIE stellt sich die Frage? ABAP hat **zwei Möglichkeiten**:

* Das ist ein Kommentar, aber nur wenn der

* (Stern) in der ersten Spalte steht

```
WRITE / 'zum Beispiel: ABAP ich träum, von dir!' "in Zeile
```

Was manchmal langweilig sein kann, ist, dass man **Coding-Zeilen** mit identischem Beginn schreiben muss, und das mehrfach hintereinander. Da gibt es eine **Variante**, die sich **Kettensatz** nennt, das heißt, schreibe das **Schlüsselwort** einmal gefolgt von einem **Doppelpunkt**, und trenne die Zusätze durch **Kommata**:

```
WRITE: 'Und nun', 'ein weiteres', 'pädagogisch wertvolles', 'Beispiel'.
```

Aber so etwas geht auch mit einem Kettensatz:

```
DATA: gd_count TYPE i, gd_first_snack TYPE string value 'Buchteln'.
```

Einstieg in die **Modularisierung** the easy way: In ABAP kannst du ein ausführbares Programm (den Programmtyp verwenden wir für unser Beispiel) mithilfe von **Ereignisschlüsselwörtern** untergliedern.

Zu bestimmten **Zeitpunkten** werden die so markierten Blöcke automatisch vom **Laufzeitsystem** ausgeführt, man könnte sogar von angesprungen sprechen. Ein Ereignisschlüsselwort markiert die Sprungmarke, und von dort weg wird der Programmcode ausgeführt. Das Standardereignis ist **START-OF-SELECTION**.

Und nun bist du dran.

Ich lasse dich einfach mal probieren.

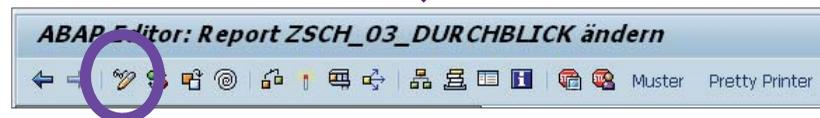
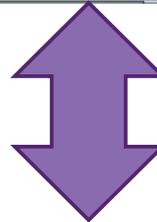
Der Durchblicker – Programm entwickeln

Du darfst ein Programm in Raten schreiben, das alle Daten von der Datenbanktabelle **ZSCH03PROJECT** liest und in einer Liste ausgibt. Hier ist die Anleitung dazu:

1. Du bist noch im ABAP Editor und musst darauf achten, dass du im **Änderungsmodus** bist, also nicht im **Anzeigemodus**. Das kannst du ganz einfach feststellen, indem du den **Titel der Seite** liest.



Hier steht's



Anzeigemodus, Änderungsmodus. Mit dem Bleistiftsymbol (Str + F1) kannst du zwischen den Modi hin- und herspringen.

2. Nun platzierst du den Cursor im ABAP-Texteditor, und schon beginnt deine Programmiererkarriere.
3. Deklariere die Variable **gs_project** vom Typ **zsch03project**.
4. Platziere darunter das Ereignis **START-OF-SELECTION**.
5. Schreibe nach dem **START-OF-SELECTION** mithilfe des **WRITE**-Statements den Text **'Durchblick 2.0'**.

Das ist alles?

Ja, das ist alles! Und wenn du das geschafft hast, kannst du dir wieder die folgende Sequenz an Aktionen verinnerlichen.



1. Sichern des Quelltextes mit der **Sichern**-Drucktaste (Strg + S) in der Symbolleiste.
2. Syntaxchecker anwerfen mit der **Prüfen**-Drucktaste (Strg + F2) (die mit dem Waagesymbol) in der Drucktastenleiste.
3. Aktivieren mit der **Aktivieren**-Drucktaste (Strg + F 3) (die mit dem Zündholz) in der Drucktastenleiste.
4. Testen mit der **Direkt**-Drucktaste (F8) (die mit dem Schublehresymbol, könnte aber auch eine Schraubzwinde sein).

Also leg los, ich warte mal ... dumdideldum ... heiapopeia ... lalala ...



Fertig!

Und, was siehst du? Hast du den Durchblick? Ich habe die Aufgabenstellung so umgesetzt:

```
REPORT zsch_03_durchblick.
* Die Variable zum Befüllen
DATA: gs_project TYPE zsch03project.
* Hier springt die Laufzeitumgebung rein
START-OF-SELECTION.
* Das mächtige WRITE zaubert eine Zeile in die Liste
WRITE: / 'Durchblick 2.0'.
```

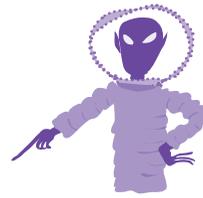


Wunderschöne
Liste mit deiner
Ausgabe.
Willkommen
im Club.

Falls du noch Fragen zum ABAP Editor hast,
dann lies die Doku!
Oder noch besser Kapitel 4.

*jetzt hab ich
Lunte gerochen, ich will Daten
von der Datenbank lesen!*

Her mit den Daten – DB-Zugriff



TATENFERARPEIDUNK ist das, was wir wollen!

In ABAP ist es sehr einfach, mit der Datenbank ins Gespräch zu kommen.

In die Sprache ABAP ist die **Open-SQL-Variante von Standard-ANSI-SQL** eingebaut, die unter anderem das für dich wichtigste Schlüsselwort zur Verfügung stellt:

Keyword	Function
SELECT	Reads data from database tables.

Open SQL arbeitet mit **Tabellen**, die über das Dictionary definiert werden. Das kennst du schon.

Falls du nun einen Eintrag aus einer Tabelle lesen möchtest, verwendest du die SELECT-Anweisungen. Dies könnte für unser Beispiel so aussehen.

```
SELECT SINGLE * *1 FROM zsch03project *2 INTO gs_project *3 WHERE projekt = '001' *4.
```

***1** Zuerst gibst du die **Ergebnismenge** an, wobei * bedeutet, dass du alle Spalten der Tabelle lesen möchtest. Falls du nur an einer Zeile der Datenbanktabelle interessiert bist, verwende ein **SINGLE** nach dem **SELECT**.

***2** Hier legst du fest, woher die Daten kommen. Das kann zum Beispiel deine **Projekttable** sein.

***3** Irgendwo im Programm müssen die Daten gespeichert werden. Wo, das gibst du hier an. Wir verwenden natürlich die Variable, die wir vorher angelegt haben. Übrigens wird die Variable auch als **Struktur** bezeichnet, da **mehrere Felder** in ihr stecken.

***4** Die Treffermenge kann mithilfe der **WHERE**-Bedingung eingeschränkt werden, also weniger Zeilen im Ergebnis bewirken. Wir lesen nur den Eintrag zum Projekt **001**.

Weil es einfach hierher passt, noch ein paar Zusatzinformationen.

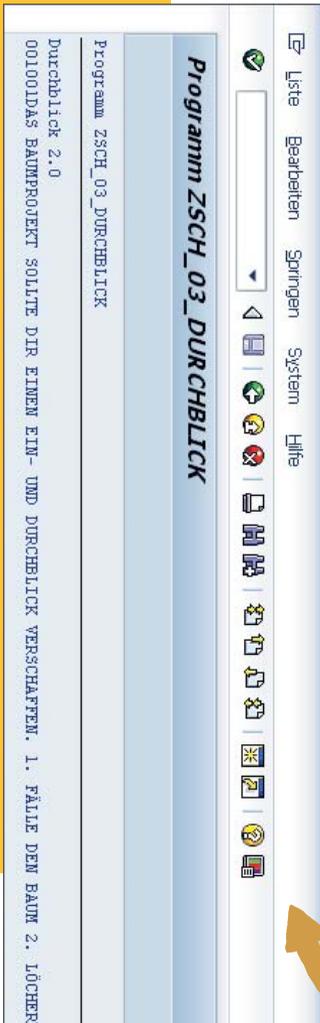
Die brauchen wir dann gleich.

Wie kannst du feststellen, ob die **Leseoperation erfolgreich** war oder nicht?

Also zuerst liest du mal die Dokumentation (Cursor auf **SELECT** und F1-Taste drücken), und dann wirst du feststellen, dass durch den Aufruf des **SELECT**-Statements die Werte von zwei **Systemvariablen** gesetzt werden: **sy-subrc** und **sy-dbcnt**.

Das Feld **sy-subrc** stellt das Ergebnis der **SELECT**-Operation in Form eines Wertes dar. Vereinfacht gesagt: Falls **sy-subrc** den Wert 0 hat, dann war alles in Ordnung, und ein Ergebnis liegt vor. Falls der Wert ungleich 0 ist, ist etwas schiefgelaufen, oder du hast keine Daten gefunden. Falls Daten im DB-System gefunden wurden, wird die Anzahl der gefundenen Datensätze in das Feld **sy-dbcnt** gestellt.

Jetzt lese ich



Ein Eintrag aus der Datenbanktabelle in deiner Liste. Damit kommunizierst du vom Programm aus mit der Datenbank.

Du bist an der Reihe mit dem Lesen von der Datenbank.

[Einfache Aufgabe]

Lies den Eintrag für das Projekt **001** von der Datenbanktabelle **zsch03project** in die Zielstruktur **gs_project**. Gib direkt danach die Struktur mit einem **WRITE: / gs_project** aus. Was macht dieses komische / hier?

Also: Es gibt schwere Aufgaben, es gibt einfache Aufgaben, und es gibt ... Aufgaben. Das war ja total einfach! Und das / bedeutet Zeilenumbruch!

Das liegt natürlich an meinem ausgefeilten pädagogischen Konzept, an deiner zwölften Kanne Kaffee und daran, dass du so ein cleveres Kerlchen bist. Zeig mal deine Lösung.

Und die Ausgabe sieht so aus:

```
REPORT zsch_03_durchblick.

DATA: gs_project TYPE zsch03project.

START-OF-SELECTION.
  WRITE: / 'Durchblick 2.0'.
  * Einzelsatz lesen
  SELECT SINGLE * FROM zsch03project INTO gs_project
    WHERE projekt = '001'.
  * und in der Liste ausgeben
  WRITE: / gs_project.
```

Hast du dir möglicherweise schon überlegt, wie wir darauf reagieren können, dass keine Daten in der Datenbanktabelle gefunden werden können? Das könntest du in deiner Pause machen.

```
* Einzelsatz lesen
SELECT SINGLE * FROM zsch03project INTO gs_project
  WHERE projekt = '001'.
IF sy-subrc = 0.
* und in der Liste ausgeben
  WRITE: / gs_project.
ELSE.
  WRITE: / 'Nix da, leider!'.
ENDIF.
```

Ohne Struktur läuft nichts – Grundlegende Programmstruktur

Ein wenig Logik schadet niemandem, auch wenn man sehr stark „Bauchmensch“ ist. Obwohl, da habe ich dieses spannende Buch über „Weniger Kopf mehr Bauch“ und dann wieder „Kopf schlägt Kapital“, irgendwie ist immer **Logik** dabei. Also, was brauchen wir für den logischen Beginn? Möglicherweise eine **Möglichkeit**, mithilfe des **IF**-Statements zu entscheiden.

```
IF <irgendwas>. "ja, das ist ein Punkt  
* Dann mach etwas  
ELSE. "Alternative  
* Dann mach etwas Anderes  
ENDIF. "Ende mit der Logik
```

Das **<irgendwas>** im **IF** ist interessant. Ich werde dazu **logische Bedingung** sagen. Du kannst zum Beispiel Vergleiche anstellen, wie **IF gs_project-projekt = '001'**.

*Halt, Roland.
Was ist denn mit dem Bindestrich?
Das gs_project ist die Struktur
von vashin.*

Okay, zu schnell. Die Struktur ist ja aufgebaut wie die transparente Tabelle **zsch03project**, und die hat unterschiedliche Spalten. Stell dir einfach die Struktur wie eine Zeile aus der Tabelle vor. Damit du auf die einzelnen Spalten, ich meine Felder, zugreifen kannst, musst du das dem System irgendwie sagen. Darum der **Bindestrich**, weil du damit sagen kannst, dass du am Feld **projekt** in der Struktur **gs_project** interessiert bist.

Als Alternative zum **IF**-Statement kannst du auch das **CASE**-Statement verwenden, das in seiner Grundform folgendermaßen aufgebaut ist:

```

CASE <variable>.
WHEN <Wert>.
* Mach was für den ersten Wert
WHEN OTHERS.
* Mach was sonst
ENDCASE.

```

Kurzer Fokus: Prüfe nach dem **SELECT SINGLE**, ob ein Wert in der Datenbanktabelle gefunden wurde. Verwende dazu **IF, ELSE, ENDIF, =, sy-subrc**, den Wert **0**. Dieses Mal habe ich einen Lückentext für dich.

```

REPORT zsch_03_durchblick.
* Die Variable zum Befüllen
DATA: gs_project TYPE zsch03project.
* Hier springt die Laufzeitumgebung rein
START-OF-SELECTION.
* Das mächtige WRITE zaubert eine Zeile in die Liste
WRITE: / 'Durchblick 2.0'.
* Einzelsatz lesen
SELECT SINGLE * FROM zsch03project INTO gs_project
WHERE projekt = '001'.
* Jetzt auch mit logischer Kontrolle
_ _ _ _ _ = _ .
* und in der Liste ausgeben
WRITE: / gs_project.
_ _ _ _ .
* der arme Anwender
WRITE: / 'Och schade, nichts gefunden für Projekt = ',
'001'.
_ _ _ _ _ .

```

Natürlich wieder sichern, syntaxchecken und testen!
Lösung gibt es keine, da musst du schon weiterlesen.

Alles muss es auch nicht sein – Einfaches Selektionsbild

Wir lieben unsere User! Darum werden wir auch keine Mühe scheuen, ihnen das Beste vom Besten anzubieten, also zum Beispiel eine Möglichkeit, um zu bestimmen, welche Daten für sie interessant sind. SAP nennt diese Möglichkeit **Selektionsbild**. Die einfachste Möglichkeit, ein Selektionsbild zu definieren, ist das **Standardselektionsbild**.



[Achtung/Vorsicht]

Ab jetzt gilt: Wenn ich dich frage:
„Welche Nummer hat das Standardselektionsbild?“, dann musst du immer mit „1000“ antworten.

[Zettel]

Mit dieser Nummer wird ein Dynpro in einem Programm eindeutig identifiziert. Du kannst die Nummer für deine Dynpros frei vergeben, AUSSER der Nummer 1000, weil die für das Standardselektionsbild reserviert ist.

Das ist deshalb so einfach, da die Definition durch Deklaration geschieht. Das einfachste Beispiel eines **Eingabeelementes** im **Selektionsbild** ist ein **Eingabefeld** mit einem **Label**. Dieses wird als **Parameter** bezeichnet und zum Beispiel folgendermaßen deklariert:

PARAMETERS*1 pa_proj*2 TYPE*3 zsch03project-projekt*4.

*1 Mit dem deklarativen Schlüsselwort PARAMETERS wird ein Parameter deklariert,

*2 und dieser sollte auch einen **Namen** besitzen. Achtung, Softwarearchäologen: Aus historischen Gründen darf der **Name nicht länger als acht Zeichen** sein.

*4 **Der Typ**. Dadurch wird zum Beispiel die Anzahl der Zeichen bestimmt, die dem Anwender für die Eingabe zur Verfügung gestellt werden.

*3 Mit dem Schlüsselwort **TYPE** wird ein Bezug zu einem Typ hergestellt. Dieser muss **nicht nur technisch** sein, sondern kann auch **semantische Informationen** (so wie in meinem Beispiel) beinhalten, wie zum Beispiel eine **Werteilfe**. Das kommt dann alles ausführlich in Kapitel 12.

[Einfache Aufgabe]

It's exercise time!

Implementiere im Programm, direkt unter der REPORT-Zeile, den Parameter für die Eingabe des Projekts, und verwende den Zusatz OBLIGATORY am Ende.

Schon getestet? Wie sieht es aus? Steht da kein verünftiger Text für das Label zum Eingabefeld?



Nö. Ehes
abschreckend.

Das habe ich mir gedacht.

Das **Label** zum **Eingabefeld** kann über das Menü **Springen • Textelemente • Selektionstexte** definiert werden. Bevor du zu springen anfängst, musst du unbedingt „Aktivieren“ (Zündholz in der Drucktastenleiste). Übrigens ist der Parameter im Programm wie eine Variable verwendbar.



[Einfache Aufgabe]

Springe zu den Selektionstexten, tippe direkt neben dem Parameter PA_PROJ, dort, wo das Fragezeichen steht, den Text „Das Projekt“ ein, und aktiviere den Text mit dem Aktivieren-Button.

Wechsle danach wieder zurück (F3), und starte das Programm.

Bei mir sieht das folgendermaßen aus:

```
REPORT zsch_03_durchblick.
```

```
* Parameter für das Projekt
```

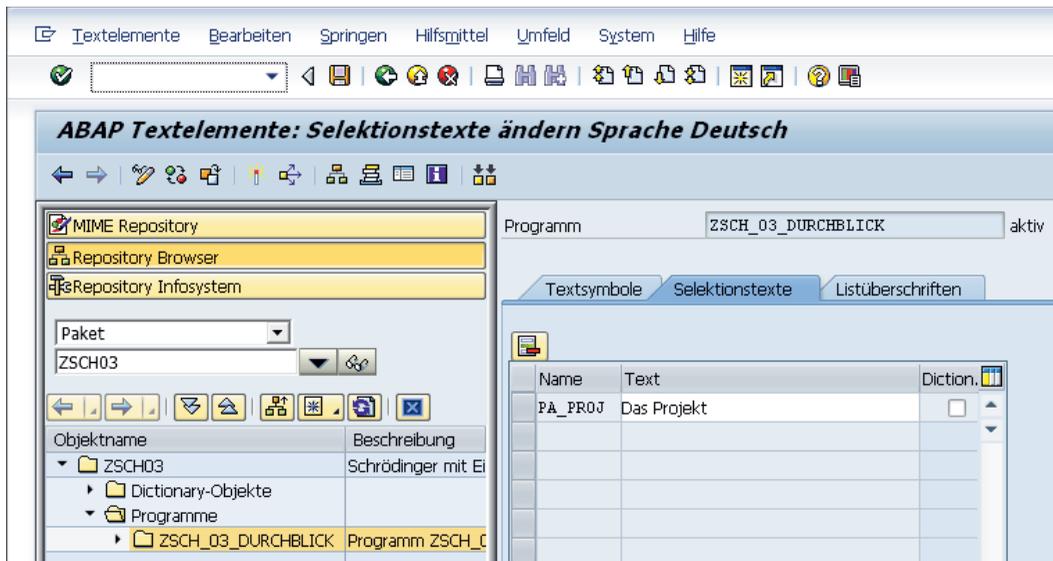
```
PARAMETERS: pa_proj TYPE zsch03project-projekt OBLIGATORY.*1
```

```
* Die Variable zum Befüllen
```

```
DATA: gs_project TYPE zsch03project.
```

*1 Da ist der Parameter aber schön zwischen **REPORT** und **DATA** eingeklemmt. Der neue Teil ist der Zusatz **OBLIGATORY**. Der bewirkt, dass der Anwender einen Wert eingeben **MUSS**. Falls der Anwender trotzdem das Programm ohne starten möchte, wird er mit Fehlermeldungen überflutet.

Dann ist noch der Selektionstext zu definieren:

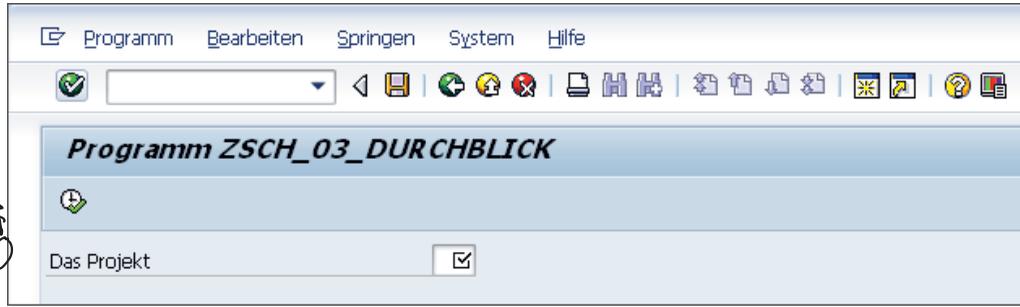


Selektionstextpflege. Sieht doch viel besser aus als PA_PROJ im Selektionsbild.





Und das wunderschöne Ergebnis:

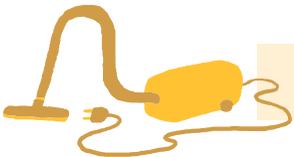


„Das Projekt“ als Label für das Eingabefeld klingt wie der nächste Roman von John Grisham.



*Alles super, alles paletti.
Dennoch ist etwas eigenartig. Egal,
welchen Wert ich eingabe und das
Programm mit dem grünen Häkchen (F8)
starte, ich bekomme immer denselben
Datensatz. Super Selektionsbild*

Ahhh, da fehlt noch was. Du hast Recht. Natürlich müssen wir den Parameter bei der Selektion berücksichtigen. Oben habe ich erwähnt, dass du den Parameter wie eine Variable im Programm verwenden kannst.



[Einfache Aufgabe]

Ersetze im Programm zsch_03_durchblick alle Vorkommen von ‚001‘ durch pa_proj.
Und teste dein Programm!

Das Entsetzen kannst du am einfachsten mit Strg + F bewerkstelligen.
Nach der Ersetzung sollte der vollständige Text so aussehen:

```
REPORT zsch_03_durchblick.  
* Parameter für das Projekt  
PARAMETERS: pa_proj TYPE zsch03project-projekt OBLIGATORY.  
* Die Variable zum Befüllen  
DATA: gs_project TYPE zsch03project.  
* Hier springt die Laufzeitumgebung rein  
START-OF-SELECTION.  
* Das mächtige WRITE zaubert eine Zeile in die Liste
```

```

WRITE: / 'Durchblick 2.0'.
* Einzelsatz lesen
SELECT SINGLE * FROM zsch03project INTO gs_project
  WHERE projekt = pa_proj.
* Jetzt auch mit logischer Kontrolle
IF sy-subrc = 0.
* und in der Liste ausgeben
  WRITE: / gs_project.
ELSE.
* der arme Anwender
  WRITE: / 'Och schade, nichts gefunden für Projekt = ', pa_proj.
ENDIF.

```

Was geschieht beim Testen? Bei mir passiert Folgendes:

1. Ich gebe im Selektionsbild keinen Wert ein, und starte das Programm. Eeeeeeh. Fehlermeldung: „Füllen Sie alle Mussfelder aus.“
2. Ich gebe die Nummer 001 ein. Ahhhh, wunderschöne Liste mit Werten.
3. Ich gebe die Nummer 002 ein. Eeeeeh, die Liste wird angezeigt, aber mit einer Meldung, dass für den Datensatz 002 kein Projekt gefunden wurde.

Eine etwas komplexere Form der Eingabe wird mit **SELECT-OPTIONS** definiert. Dabei werden Intervalle von Werten für die Eingabe angeboten. Das ist aber für den Augenblick zu kompliziert und wird daher auf Kapitel 12 verschoben.

[Belohnung/Lösung]

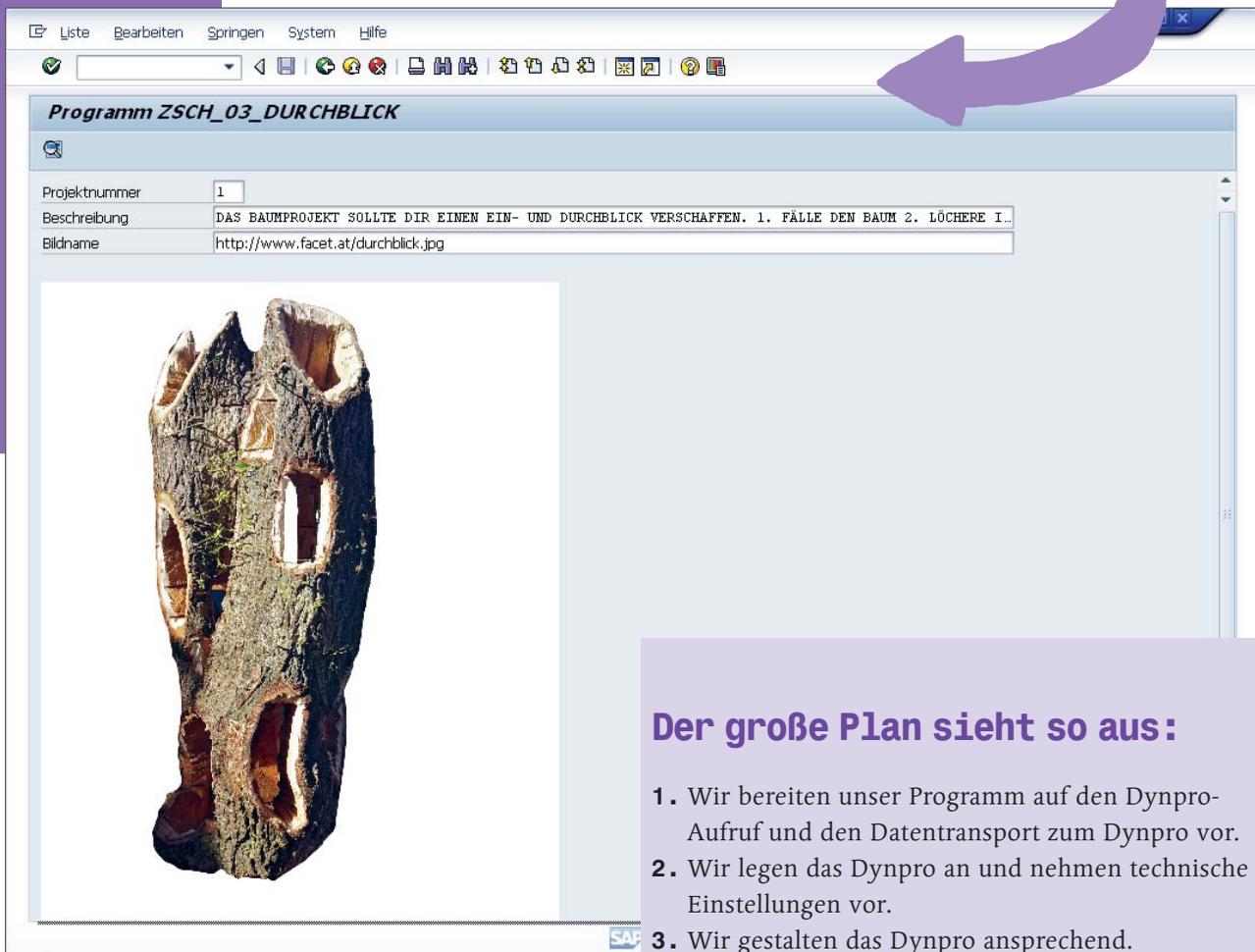
Apropos kompliziert und verschoben. Hast du schon mal versucht, bei dir in der Wohnung den Kasten dort hinzuschieben, wo der Stuhl war, und diesen im Vorraum direkt neben dem Kleiderhaken aufzuhängen? Dann könntest du nämlich endlich Platz schaffen für das Sofa, das eigentlich auf den Platz gehört, wo jetzt der Tisch steht.

Bitte ein Vorher-/Nachher-Foto an die Redaktion. Danke!

Layouten und Daten transportieren – Ein einfaches Dynpro

Wenn ein Abschnitt die Bezeichnung „Werkstatt“ verdient, dann ist es dieser. Du wirst schrauben und hämmern, sägen und kleben. Nur eben digital.

Ein **Dynpro (dynamisches Programm)** ist ein Bild mit einer sogenannten Ablauflogik, also mit einem Programm, und wenn du willst, kannst du es mit dem Selektionsbild vergleichen. Halt, da muss ich etwas präziser sein. **Ein Selektionsbild ist ein Spezialfall eines Dynpros.** Wir werden ein Dynpro verwenden, um die von der Datenbank selektierten Daten darzustellen, sogar mit Bild! Schauen wir uns das mal Schritt für Schritt an. **Am Ende kommt das raus:**



Projektnummer	1
Beschreibung	DAS BAUMPROJEKT SOLLTE DIR EINEN EIN- UND DURCHBLICK VERSCHAFFEN. 1. FÄLLE DEN BAUM 2. LÖCHERE I...
Bildname	http://www.facet.at/durchblick.jpg



Wow, damit hat man ja richtig den Durchblick in SAP.

Der große Plan sieht so aus:

1. Wir bereiten unser Programm auf den Dynpro-Aufruf und den Datentransport zum Dynpro vor.
2. Wir legen das Dynpro an und nehmen technische Einstellungen vor.
3. Wir gestalten das Dynpro ansprechend.
4. Wir programmieren die Darstellung des Bildes mithilfe der Objektorientierung.

Und nun Schritt für Schritt, im Sinn von Adam Ries: Mach so, und es ist richtig:

Im Programm haben wir die folgenden Vorbereitungen zu treffen:

```
REPORT zsch_03_durchblick.
```

* TABLES-Struktur für Dynpro-Daten

```
TABLES: zsch03project.*1
```

* Parameter für das Projekt

```
PARAMETERS: pa_proj TYPE zsch03project-projekt OBLIGATORY.
```

* Die Variable zum Befüllen

```
DATA: gs_project TYPE zsch03project.
```

* Controls *2

```
DATA: gr_container TYPE REF TO cl_gui_custom_container,  
      gr_picture TYPE REF TO cl_gui_picture.
```

* Hier springt die Laufzeitumgebung rein

```
START-OF-SELECTION.
```

* Das mächtige WRITE zaubert eine Zeile in die Liste

```
WRITE: / 'Durchblick 2.0'.
```

* Einzelsatz lesen

```
SELECT SINGLE * FROM zsch03project INTO gs_project  
      WHERE projekt = pa_proj.
```

* Jetzt auch mit logischer Kontrolle

```
IF sy-subrc = 0.
```

* und in der Liste ausgeben

```
WRITE: / gs_project.  
ELSE.
```

* der arme Anwender

```
WRITE: / 'Och schade, nichts gefunden für Projekt = ', pa_proj.  
ENDIF.
```

* Hier springt die Laufzeitumgebung rein

```
AT LINE-SELECTION.*3
```

* Daten in die TABLES-Struktur

```
zsch03project = gs_project.*4
```

* Dynpro aufrufen

```
CALL SCREEN 9100.*5
```

*1 Füge direkt unterhalb von **REPORT** die Anweisung **TABLES: zsch03project** ein. Der Effekt der **TABLES**-Anweisung ist vergleichbar mit der **DATA**-Anweisung, somit legst du damit eine Variable mit dem Namen **zsch03project** an. Ich nenne sie **Kommunikationsstruktur**, weil sie zur Datenkommunikation zwischen Programm und Dynpro dient.

Wofür?

Diese Variable dient uns zum Datenaustausch mit dem Dynpro. Sonst sehen wir ja nix.

*2 Der Controls-Abschnitt, so wie ich ihn genannt habe, enthält zwei Variablen. Die zwei Objektreferenzen für die Bildprogrammierung.

Ich frage jetzt mal sicherheitshalber nicht nach, was eine Objektreferenz ist, ich lasse es einfach über mich ergehen.

Auch besser so! Das erfährst du noch bis zum Abwinken in den Kapiteln 8 bis 11.

*3 Das Ereignis **AT LINE-SELECTION** wird von der Laufzeitumgebung angesprungen, falls ein Anwender doppelt in der Liste klickt. Meine Idee dahinter ist, dass der Anwender auf die Listenzeile klickt, in der der Datensatz der Datenbank-tabelle steht, und sich dann das Dynpro öffnet.

*4 Damit die Daten dem Dynpro tatsächlich zur Verfügung stehen, müssen sie in die Kommunikationsstruktur kopiert werden. Das geschieht ganz einfach durch die Zuweisung **zsch03project = gs_project**.

*5 Und endlich kommt der bombastische Aufruf **CALL SCREEN**. Damit wird ein Dynpro aufgerufen. Die Nummer ist die vierstellige Nummer des Dynpros. Ich habe 9100 gewählt, aber auf alle Fälle nicht die Nummer ...?

1000, weil das die Nummer des Standardselektionsbildes ist. Jetzt bin ich aber selber etwas baff. Also diese modernen Energy drinks ..., ob das noch legal ist?

[Einfache Aufgabe]

Implementiere die besprochenen

Änderungen im Programm. Sobald du den Befehl **CALL SCREEN 9100**. eingefügt hast, sicherst du und klickst doppelt auf **CALL SCREEN**. Und dann wartest du auf mich!

Dekorieren der Auslage – Dynpro-Layout

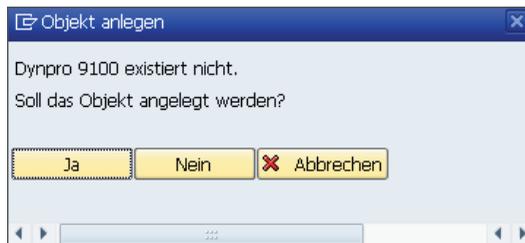
Huhu, wo bist du?
Ich habe alles so gemacht, wie du gesagt hast,
und jetzt fragt mich das System per Pop-up,
ob es irgendetwas anlegen soll.
Hinlegen statt anlegen!



Was, schon wieder fertig? Dann können wir uns jetzt dem Dynpro zuwenden.

Dynpro anlegen und technische Einstellungen vornehmen

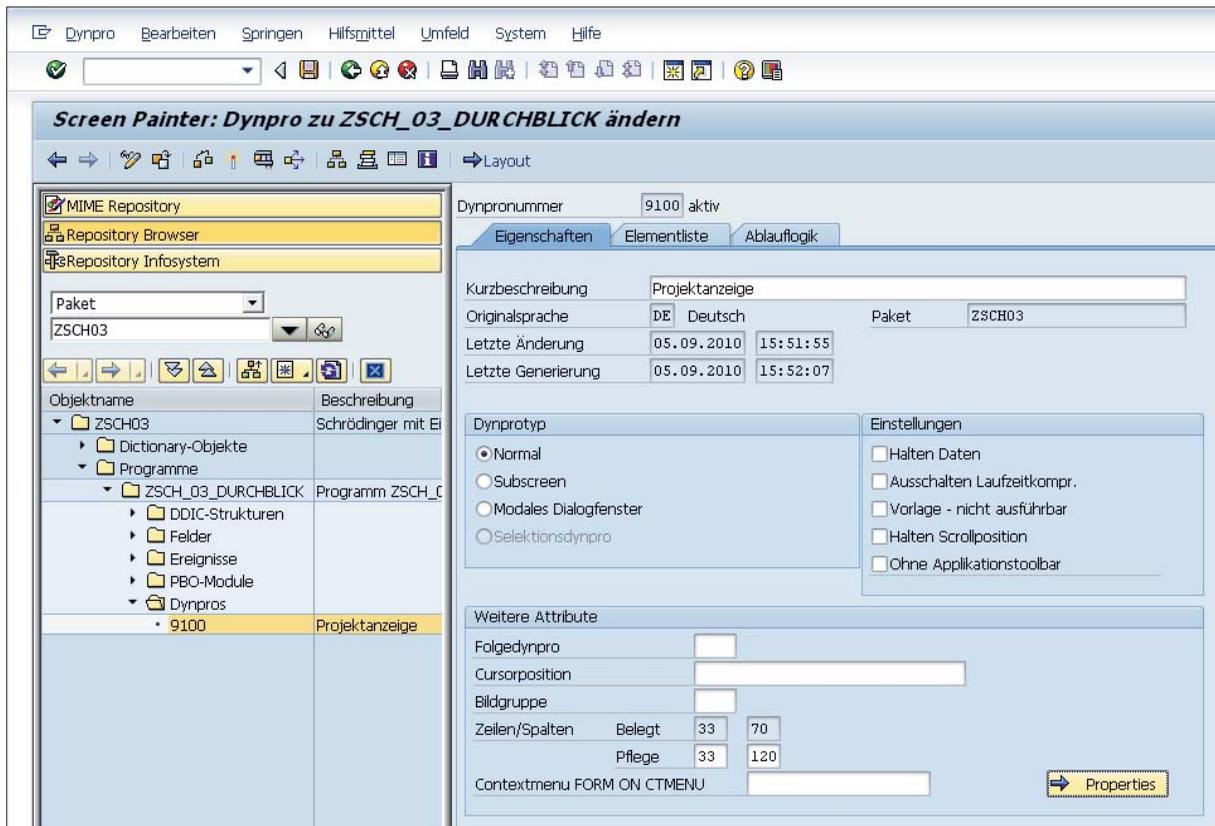
1. Du bist also schon beim Pop-up zum Anlegen des Dynpros. Das kannst du einfach mit **Ja** bestätigen, weil du ja willst, dass das Dynpro angelegt wird.



Sicher anlegen, bitte.

2. Nach deiner Bestätigung erscheint beinahe unbemerkt der sogenannte **Screen Painter** (SE51) im Werkzeugbereich. Damit werden wir ab jetzt arbeiten und Dynpros layouts und programmieren.





Der Screen Painter bietet alles, was zum Dynpro-Programmieren benötigt wird.

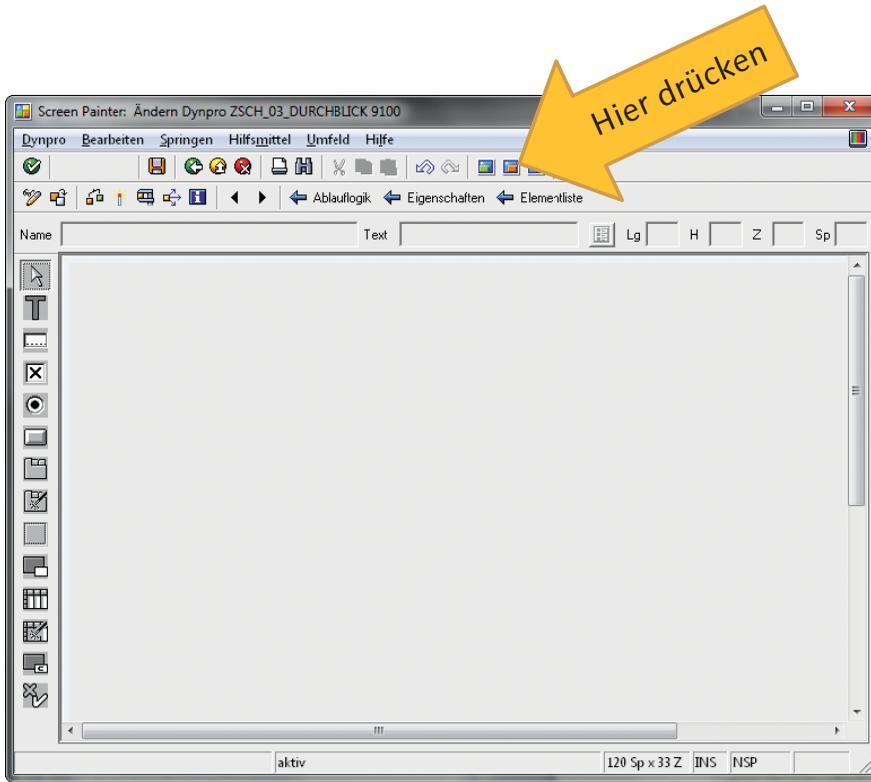
Auf dem Karteireiter **Eigenschaften** pflegst du für unser Beispiel die **Kurzbeschreibung**, die du dann in der Objektliste wiederfindest, und das **Folgedynpro**. Das steuert, wohin nach der Abarbeitung der Ablauflogik, das ist das Programm zum Dynpro, gesprungen wird. Wenn du die Nummer **9100** stehen lässt, das heißt die Nummer vom Dynpro selbst, dann kommst du dir vor wie am Kettenbrater, du bleibst auf diesem Dynpro bis zum Sankt Nimmerleinstag. Trage bitte die Nummer 0 ein, das führt dich dann automatisch zur Aufrufstelle des Dynpros zurück. Das System macht daraus ein leeres Eingabefeld. 0 = leer.

Jetzt gestalten wir das Dynpro ansprechend.

3. Unsere nächste Station ist die Gestaltung des Dynpros, das Layout.

Das Gesicht deines Programms nach außen.

Drücke bitte die Drucktaste **Layout** (Strg + F7), um in den Graphical Screen Painter zu wechseln.



So sieht ein leeres Dynpro aus, ziemlich nackt.

4. Im Layout kannst du nun gestalten. Dazu drückst du die Drucktaste **Dict-/Programmfelder-Fenster** (F6) (das ist die rote Drucktaste in der Symbolleiste), die es dir ermöglicht, auf Definitionen im ABAP Dictionary zuzugreifen. Damit wird die Gestaltung zum Kinderspiel. Tippe im erscheinenden Dialog im Eingabefeld **Tabellen-/Feldname** den Namen der transparenten Tabelle **zsch03project** ein.

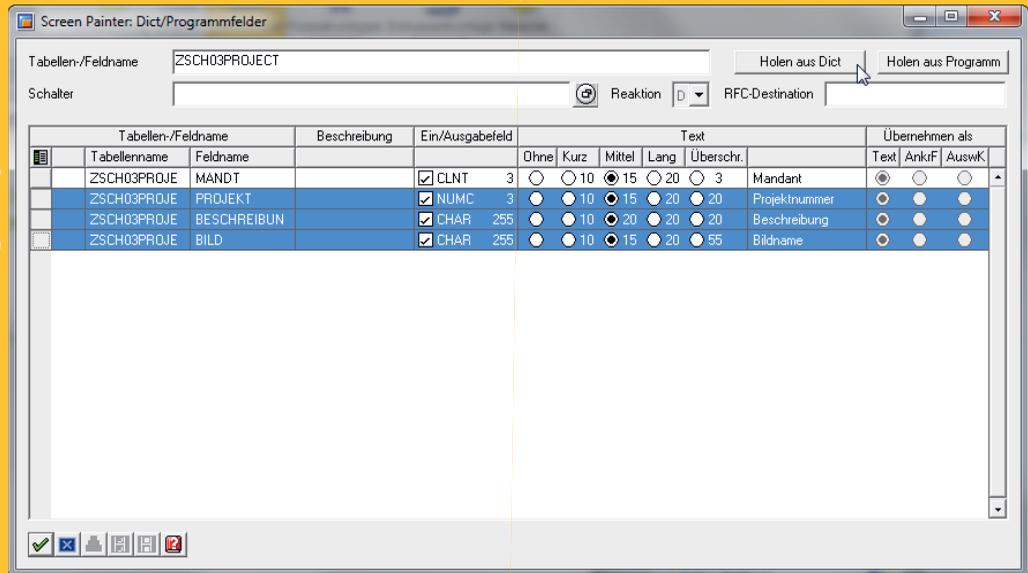


*Hey, die haben wir doch auch für die TABLES-Anweisung im Programm verwendet. Ist das Zufall? **

Natürlich nicht. Wenn wir hier dieselbe Struktur verwenden, dann transportiert das System für uns automatisch die Daten vom Dynpro in das Programm und vice versa. Namensgleichheit von Feldern im Dynpro und im Programm bedeutet, dass uns Arbeit abgenommen wird. Danke, SAP!

*** [Anmerkung des Autors: Da hab ich dem Schrödinger aber eine blöde Frage in den Mund gelegt!]**

Tabelle
zsch03project
 eingeben, die
 Drucktaste **Holen
 aus Dictionary**
 drücken und die
 gewünschten **Felder
 markieren**, die im
 Layout erscheinen
 sollen. Bestätigen mit
 dem grünen Häkchen.
 Fertig.



Dann die Drucktaste **Holen aus Dictionary** drücken und aus der erscheinenden Liste von Feldern die für das Layout gewünschten auswählen. Wie wirst du wohl deine Auswahl bestätigen können?

- A grüner Haken
- B blaues X
- C rotes Fragezeichen

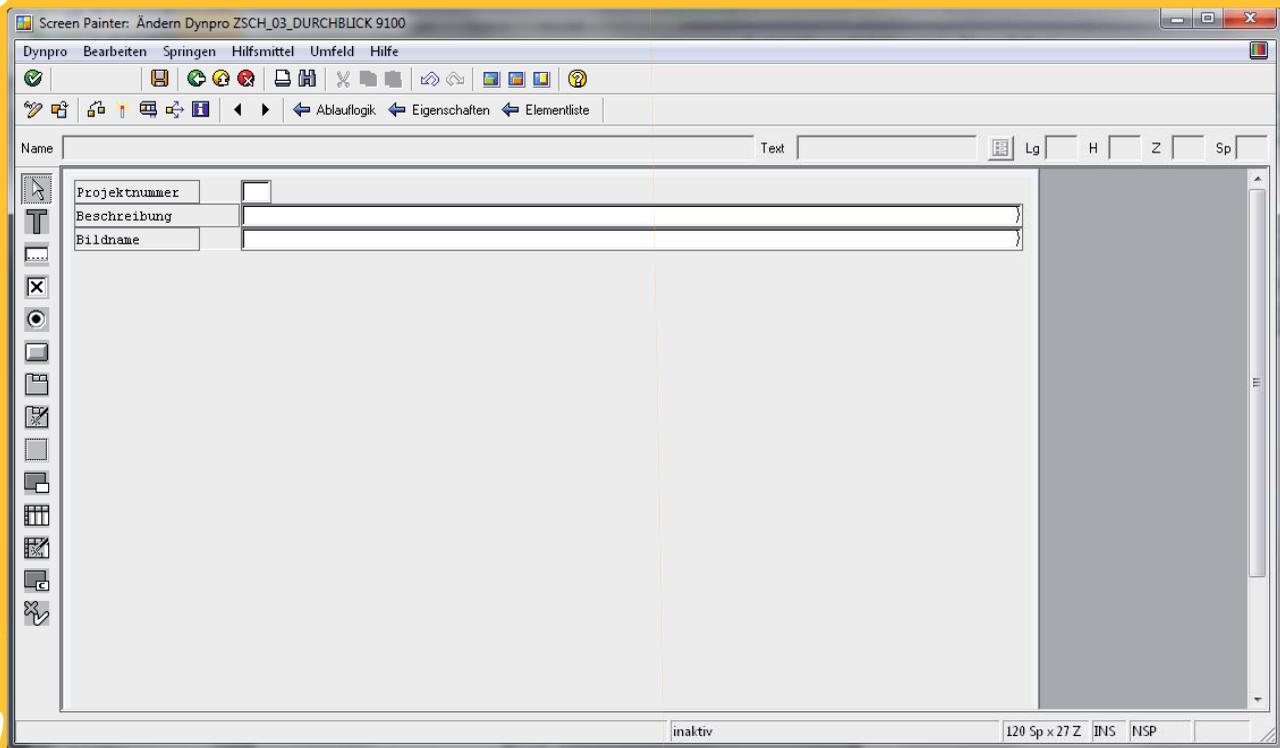
Sobald du bestätigt hast und mit dem Cursor wieder in das Layout fährst, siehst du schon das Geisterbild der drei Felder – **Huahhhh.**



[Achtung/Vorsicht]

Es hat sich beim Buchdruck herausgestellt, dass Geisterbilder nur von Mitternacht bis 01:00 Uhr im Buch sichtbar sind, darum hat man von einem Abdruck der Geisterbilder abgesehen.

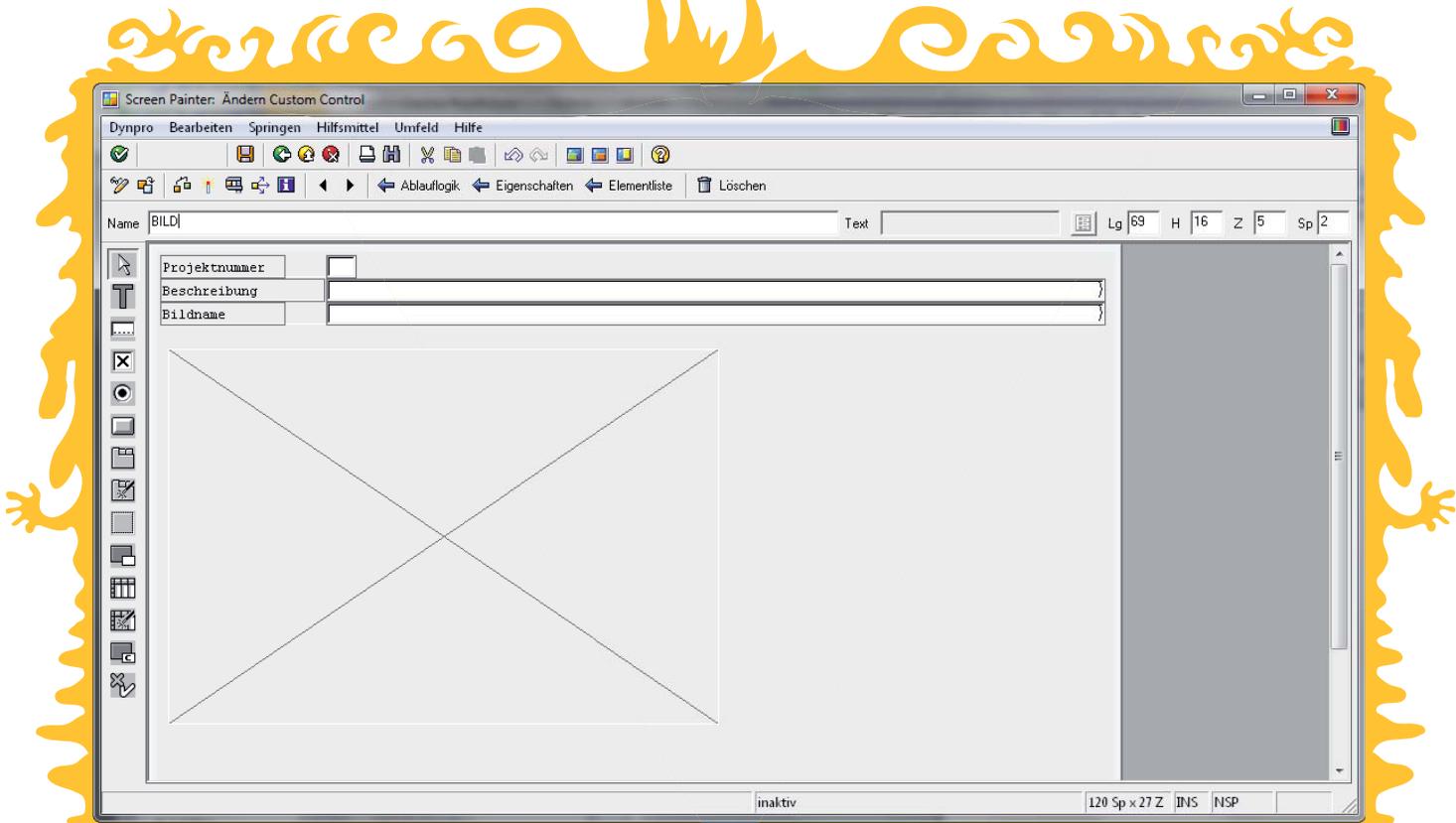
Platziere die drei Felder ganz links oben auf dem Dynpro, indem du dorthin klickst. Plumps, die Felder sind platziert, und du erhältst noch eine Warnung, die dich nicht stören sollte. Drück sie einfach weg, und denke dir: Mir egal!



Drei Föda a gführiger Schnee juchee, des is doch mein gresde Gaude. (Übersetzung entweder bekannt aus dem Winterurlaub oder anforderbar beim Verlag.)

Spitze, da sind ja unsere Felder. Wenn du willst, kannst du noch die Länge verändern, indem du an den rechten Rand klickst und ihn links rüberziehst. Das ist aber eine **Fleißübung**.

5. Letzter Schritt in der Gestaltung: Du reservierst einen Bereich für die Darstellung des Bildes. Dazu klickst du aus der Werkzeugliste das Symbol mit dem **C** (das vorletzte) an. Das C steht jetzt nicht für Chaos, Vitamin C oder Kreuzworträtsel, sondern für **Custom Control**. Also werden in diesem Bereich visuelle Controls eingeblendet. Um den Bereich nun im Layout festzulegen, markierst du die gewünschte linke obere Ecke durch einen Klick, hältst weiter die Maustaste gedrückt und ziehst zur gewünschten rechten unteren Ecke. Kann man verstehen, was ich gerade geschrieben habe?



Das große X markiert den reservierten Bereich, und einen **Namen** hat der Bereich auch schon bekommen, nämlich **BILD**.

Du gibst dem Bereich noch den Namen **BILD** im Eingabefeld **Name** und sicherst deine Eingabe.

6. Wieder zurück mit der Zurück-Taste (grüner Pfeil) oder F3.

Ablauflogik ohne Ablaufdatum – Ablauflogik programmieren



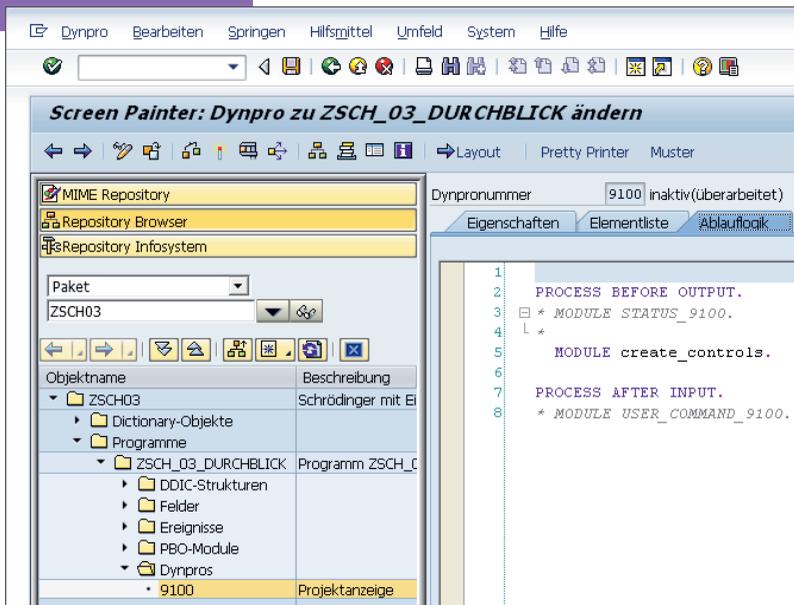
Die Gestaltung haben wir hinter uns gebracht, und uns trennen nur mehr ein paar Schritte vom erfolgreichen Abschluss des Meisterwerks. Die Ablauflogik will programmiert werden.

Jetzt programmieren wir das Programm zur Darstellung des Bildes.

[Notiz]

Die Ablauflogik eines Dynpros besteht im Allgemeinen aus zwei Teilen. Dem Teil, der ausgeführt wird, bevor das Layout angezeigt wird. Dieser nennt sich Process Before Output oder besser bekannt unter PBO.

Nachdem der Anwender etwas im Layout getan hat, wird der zweite Block ausgeführt – Process After Input oder auch PAI. Beide Blöcke können mithilfe von Modulen programmiert werden.



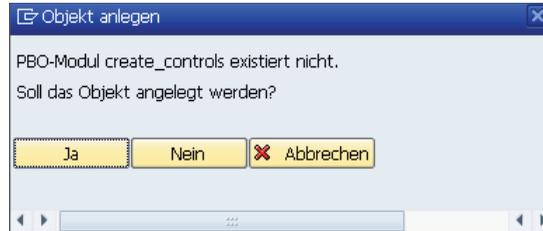
1. Wechsle bitte auf den Karteireiter **Ablauflogik** im Screen Painter.

Dein Beitrag zur Ablauflogik.

Dort siehst du zwei aktive Ereignisse, **PROCESS BEFORE OUTPUT** (PBO) und **PROCESS AFTER INPUT** (PAI). Füge die Anweisung **MODULE create_controls** zwischen PBO und PAI ein, somit wird dieses Modul zum Zeitpunkt PBO aufgerufen. Und nein, du musst mich nicht fragen, ich verrate dir sicher nicht, dass **MODULE** ein Befehl aus dem **Dynpro-ABAP**-Wortschatz ist. Das verrate ich dir erst in Kapitel 13.

Jetzt steht es drin, und nun?

2. Doppelklick auf den Modulnamen **create_controls**. Dadurch wird das Modul angelegt. Du musst im folgenden Pop-up nur **Ja** sagen.



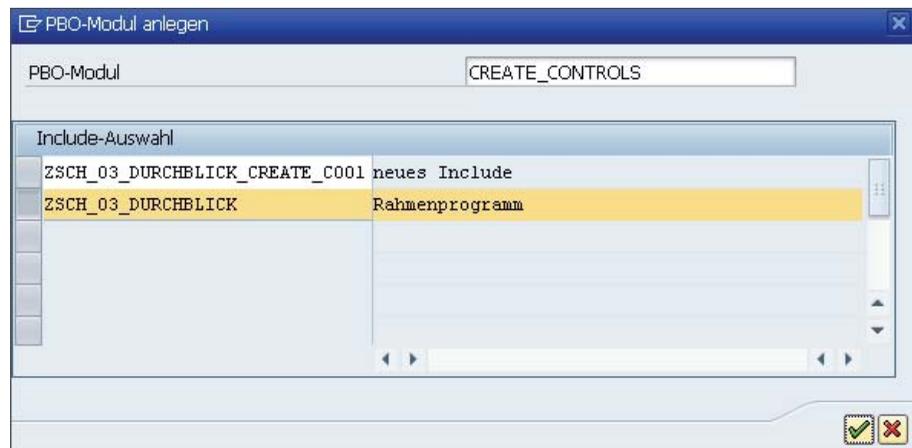
Sicher anlegen. Her damit.

3. Der nächste Dialog fragt dich, wohin das Modul gestellt werden soll.

Entschuldigung, wir hätten da so ein Modul abzubriefen. Wo sollen wir es denn hinstellen?



Wir nehmen die einfachste Möglichkeit, im **Rahmenprogramm** anlegen. Du musst nur die Zeile mit dem Rahmenprogramm auswählen und bestätigen, und schon wird dein Modul **create_controls** im Rahmenprogramm erzeugt.



Wir nehmen das **Rahmenprogramm**, sonst ist so viel zu tun und zu erklären. Es ist schon ziemlich spät.

4. Im Modul baust du bitte folgende Anweisungen ein, die ich dir kurz erklären werde:

```
MODULE create_controls OUTPUT.
```

```
* Control instanzieren
```

```
IF gr_container IS NOT BOUND.*1
```

```
* Container
```

```
CREATE OBJECT gr_container*2
```

```
EXPORTING
```

```
container_name = 'BILD'.
```

```
* Bild
```

```
CREATE OBJECT gr_picture*3
```

```
EXPORTING
```

```
parent = gr_container.
```

```
* Bild laden
```

```
CALL METHOD gr_picture->load_picture_from_url*4
```

```
EXPORTING
```

```
url = gs_project-bild.
```

```
ENDIF.
```

```
ENDMODULE.
```

*1 Die Darstellung des Bildes baut auf dem sogenannten **Control Framework** auf. Das ist eine Sammlung von objektorientierten Klassen, die für die GUI-Programmierung gedacht sind. Die Klassen bieten dir Funktionen an. Damit du diese nutzen kannst, musst du Objekte von den Klassen erzeugen. Ein Objekt ist wie eine Variable, nur eben mehr. (Kapitel zur OO lesen, dann lichtet sich der Schleier.)

Wir verwenden zwei Objekte: eines für das **Bild (gr_picture)** und eines, das sich um den Rest kümmert, wie die **Verwaltung** der Darstellungsfläche (**gr_container**).

Um festzustellen, ob das Container-Objekt schon erzeugt wurde, verwende ich **IF gr_container IS NOT BOUND.**, das frei übersetzt bedeutet: Gibt es den Container schon?

*2 Falls nicht, dann erzeuge ich ihn mit einer speziellen Funktion, **CREATE OBJECT**, und gebe den Namen **BILD** der **Custom Control Area** aus dem Layout mit. Damit weiß der Container, wofür er zuständig ist.

*3 Das Bildobjekt muss auch noch angelegt werden. Und es benötigt den Container für die Zusammenarbeit.

*4 Letzter Schritt: Wir sagen dem Bildobjekt (CALL METHOD), dass es ein Bild laden soll, das wir ihm mittels URL mitteilen. Die Info kommt aus der Datenbank.

Falls du es bis hierher geschafft hast, dann Hut ab. Du bist belastbar. Deine Belastbarkeit können wir gut für die nächsten Kapitel gebrauchen.

5. Sichern und aktivieren! Beim Aktivieren achte bitte darauf, dass du das Programm und das Dynpro aktivierst!

Achtung: Programm UND Dynpro aktivieren!

G., Objekt	Objektname	Benutzer
REPS	ZSCH_03_DURCHBLICK	BCUSER
DYNP	ZSCH_03_DURCHBLICK	9100 BCUSER



Tschinderassa-Tschniderassa- Tsching-Tsching-Tsching. Trommelwirbel

Hallo du – die Taste **Direkt** (F8) – los – drücken!

Das Selektionsbild, da gebe ich mal 001 ein.

Hallo du – die Taste **Ausführen** (F8) – los – drücken!

Die Liste mit dem Tabelleneintrag.

Hallo du – **Doppelklick** – auf die Tabellenzeile – los – jetzt!

*WOW, what a beauty. So ein schönes Bild.
Unglaublich, was man in einer Woche so alles schafft!*



Projektnummer	1
Beschreibung	DAS BAUMPROJEKT SOLLTE DIR EINEN EIN- UND DURCHBLICK VERSCHAFFEN. 1. FÄLLE DEN BAUM 2. LÖCHERE I...
Bildname	http://www.facet.at/durchblick.jpg

[Erledigt!]
Bravo, bravo, bravo. Du hast super gut durchgehalten und wurdest auch entsprechend belohnt. Bevor du nun vor lauter Erschöpfung in Ohnmacht fällst, würde ich mit dir noch gerne eine Transaktion für dein Programm anlegen. Dann ist es rund.

Na gut, dann komme ich nochmal aus dem Saug.



Das hast du dir verdient – nochmals das Bild zu sehen. Jetzt verbindest du wohl etwas anderes damit als am Anfang.

Ein Shortcut für den User – Transaktionscode anlegen/ausführen

Die **Transaktion** (das ist eine ziemlich verwirrende Bezeichnung für einen Shortcut) ist der wohlverdiente Abschluss der Entwicklung. Damit denken wir, so wie natürlich immer, an den **Komfort des Anwenders**, da dieser die Transaktion aufrufen wird, um in den Genuss der von uns entwickelten Funktionalität zu kommen.

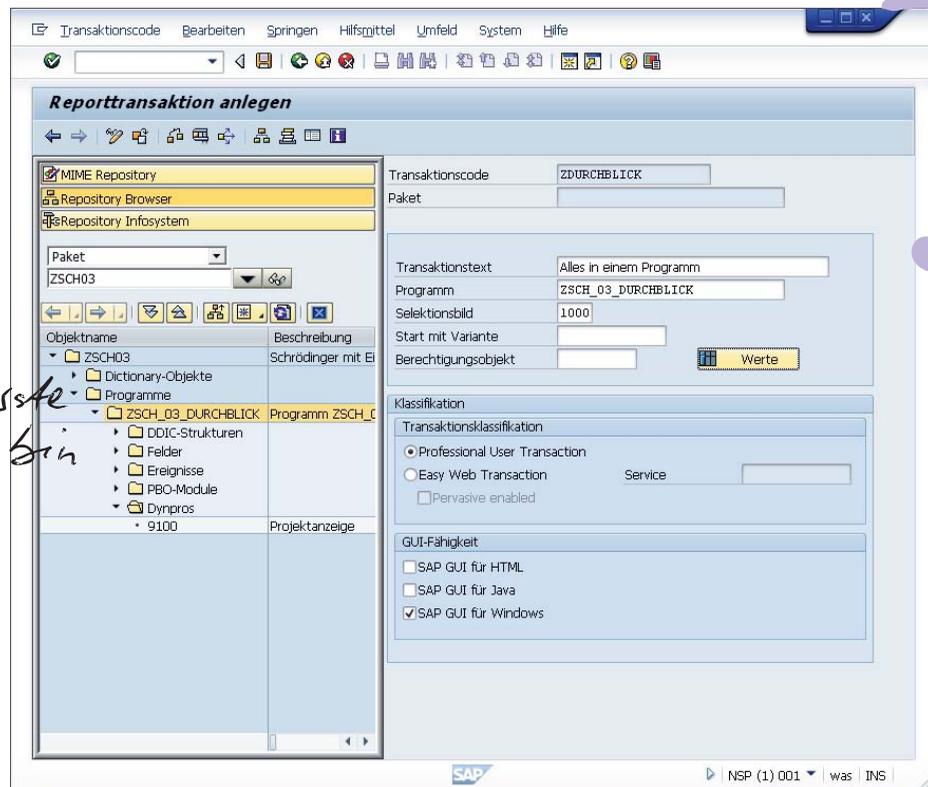
Das Anlegen einer Transaktion ist wie so vieles andere auch am einfachsten über das Kontextmenü des Programms in der Objektliste zu erledigen. Der Menüpunkt **Anlegen • Transaktion** öffnet den Pflegedialog.



Ein heikler Punkt ist sicher die Klassifikation der GUI-Fähigkeit, in unserem Fall ist „**SAP GUI für Windows**“ angebracht – noch ist das Web nicht dran ;-)

Im Feld **Transaktionscode** legst du den Namen fest, mit dem der Anwender das Programm starten kann. Als Startobjekt haben wir ein Programm mit Selektionsbild. Bitte bestätige.

Sichern, Zuordnungen zu Paket und Änderungsauftrag vornehmen, und jetzt kommt die Nagelprobe. Tippe **/ozdurchblick** im Transaktionscode-Feld ein. Was geschieht?



Yippee, meine erste Transaktion. Ich bin glücklich.

[Erledigt!]
Und wieder geht ein Tag glücklich zu Ende.
Gute Nacht John Boy,
gute Nacht SAP.

Die Details zur Transaktion. Alles selbsterklärend und abkopierend.

Index

Symbole

- 611
- 211
->* 691
* 211
/ 211
% 611
+ 211
<\$npage>Arbeitsgebiet -> s. Nachrichtenklasse 471
<\$npage>Arbeitsstruktur -> s. Struktur 191
<\$npage>Asterix -> s. Asterisk 503
<\$npage>Beistrich -> s. Komma 480
<\$npage>Class Responsibility Collaboration -> s. CRC 273
<\$npage>Datenbanktabelle -> s. Tabelle 83
<\$npage>Dynamisches Programm -> s. Dynpro 487
<\$npage>Exemplar -> s. Objekt 282
<\$npage>Feigling -> s. Ausnahme 411
<\$npage>Form-Routine -> s. Unterprogramm 239
<\$npage>HTTP -> s. Hypertext Transfer Protocol 428
<\$npage>Ikone -> s. Icon 513
<\$npage>Instanz -> s. Objekt 282
<\$npage>JSON -> s. Java Script Object Notation 425
<\$npage>KVA -> s. Kaffeevoll-automat 268
<\$npage>Laufzeitfehler -> s. ST22 370
<\$npage>LIF -> s. Lokales Interface 381
<\$npage>Objekttyp -> s. Klasse 288
<\$npage>OMG -> s. Object Management Group 276
<\$npage>Piktogramm -> s. Icon 513
<\$npage>Programm -> s. Report 54
<\$npage>Report -> s. Programm 54

<\$npage>Runtime Type Identification -> s. RTTI 467
<\$npage>Screen -> s. Dynpro 490
<\$npage>?= -> s. Down-Cast 369
<\$npage>SE11 -> s. ABAP Dictionary 404
<\$npage>SE24 -> s. Class Builder 320
<\$npage>SE37 -> s. Function Builder 411
<\$npage>Shortcut -> s. Transaktion 120
<\$npage>Slash -> s. Schrägstrich 479
<\$npage>Textkonserve -> s. Include 95
<\$npage>UML -> s. Unified Modeling Language 276
<\$npage>URI -> s. Uniform Resource Identifier 428
<\$npage>User Interface -> s. Benutzeroberfläche 490
<\$npage>Workarea -> s. Struktur 191
\$TMP 142
1 aus n 443
4. Generation 43
7.150 Kaffeebohnen 320
32-Bit-Architektur 289
Klassendiagramm 280
==-Operator 611

A

A1S 41
abap
 Typgruppe 630
abap_ 442
ABAP 41
 Advanced Business Application Programming 41, 43
 aktuell 38
 Allgemeiner Berichtsaufbereitungs-Prozessor 435
 Anweisung 47
 Aufbau 47

aufwärtskompatibel 46
Beispiel 131
betriebswirtschaftlichen Anwendungen 46
Buchstabensalat 479
Einführung 29
Erstes Beispiel 49
Gemeinschaft 38
integriert 38
international 38
interpretierende Sprache 42
Kernthemen 50
Kompatibles 404
Makro-Assembler 42
mehrsprachige Anwendungen 46
Metaformat 42
nicht case-sensitive 97
objektorientiert 46
plattformunabhängig 46
Satz 47
Satzaufbau 198
SQL-Zugriffe 46
Syntax 96
technischer Kontext 122
typisiert 46
View 632
WRITE 96
zweiphasig 42
ABAP/3 42
ABAP/4 42
 Entstehung 43
ABAP-Crack 239
ABAP Dictionary 83, 86, 404, 458, 598
 Datenelement 178
 Datentyp 170, 178, 189
 Domäne 175
 eingebauter Typ 179
 schafft Unabhängigkeit 564
 Struktur 178
 Strukturtyp 179
 Tabellentyp 179, 189
 View 614
 vom Dynpro aus 112
 Wörterbuch 86

- ABAP-Editor 46, 95, 98, 147
 - aktivieren 149
 - Änderungsmodus 98
 - Anzeigemodus 98
 - anzeigen 149
 - Breakpoint 152
 - editieren 149
 - Einstellung 147
 - Einstellungsfenster 148
 - Feature 148
 - Funktion 149
 - Muster 149, 411
 - Pretty Printer 149
 - sichern 149
 - syntaxchecken 149
 - testen 149
- abap_false 201, 442
- ABAP-Gehirnwindung 702
- ABAP-Infrastruktur 132
- ABAP List Viewer 695
- ABAP-Masta 702
- ABAP Memory 135, 136
- ABAP Objects 44
 - Vererbung 341
- ABAP OO 30, 44, 96
- ABAP-Programm
 - Datenzugriff 597
 - Übersetzung 45
 - Variable 159
- ABAP-Programmierer
 - Die spinnen nicht, die 503
- ABAP-Schlüsselwort 47
- ABAP-Syntax 47
- abap_true 201
- ABAP-Überblick 70
- Abbruchbedingung 220
 - erreichbare 220
 - EXIT 220
- Abbruchnachricht
 - Rollback 639
- Abessinien 283
- Abgeleitete Klasse 342
- Abgeschlossenes Intervall 215
- Ablagemappe 674
- Ablageort
 - SAP-Temp 671
- Ablauflogik 108, 111, 229, 434, 487, 488, 490, 497, 531
 - Modul 236
 - PAI 236
 - programmieren 116, 502
- Abmelden 66
- Absoluter Pfad 669
- ABSTRACT 373
- Abstrakt
 - Checkbox 373
- Abstrakte Klasse 372
 - Kein Objekt instanziiieren 372
- Abzweigung 214
- ACCEPTING DUPLICATE KEYS 652
- ACID 637
- ADD 211, 218
- Addition 211
- Additives Programmieren 343
- Adressenkopie 241
- Adressierung
 - direkte 165, 288
 - indirekte 165, 288
- AFDKF 433
- Aggregation 280, 281
- Ähnlichkeit
 - LIKE 607
- Aktion 490
- Aktiv 138
- Aktivieren
 - transparente Tabelle 564
- Aktivierung 324
- Aktivierungsprotokoll 180
- Aktivitätsgruppe 550
- Aktualparameter 240, 299, 304, 535
 - Unterprogramm 247
- Aktueller Anwender 222
- Aktuelle Zeit 222
- Algorithmus
 - Laufzeit 186
- Alias 387
 - Join 633
- Alles oder nichts 383
- Allgemeinen-Berichts-Aufbereitungs-Prozessor 41, 96
- Alternativer Zweig 214
- ALV 695
- Analyse-Klassendiagramm 281
- Analysemaschine 567
- AND 212, 664
 - SELECT 611
 - View 627
- Anderes Objekt 143, 146
- Änderung
 - schreiben 638
 - selektive 656
- Änderungsauftrag 73, 74, 122
 - anlegen 75
 - Auswahl 80
 - Liste 80
 - meine Aufträge 80
 - Vorschlag 80
- Anfangswert 245, 439
- Anforderung 519
- Anforderungskomplexität 282
- Anfrage
 - DB 565
 - herstellerspezifische 565
 - Senden 429
- Anführungszeichen 239
- Ankreuzfeld 441
- Anlage
 - Funktionsgruppe 257
- Anlagetyp
 - Interface 393
- Anmeldemandant 222, 571
- Anmeldemaske 56
 - Benutzer 56
 - Kennwort 56
 - Mandant 56
 - Sprache 56
- Anmelden
 - SAP-System 36
- Anmeldesprache 222
- Anmeldesprache 39
- Anmeldung 56
- Anschwänzen 357
- ANSI SQL 595
- Anstellen 371

Anweisung
 wiederholen 217
 Anweisungsblock 215, 229
 Anweisungsfolge 229
 Anwender
 mandantenabhängiger 124
 Anwendung
 Aufbau 562
 Anwendungsdomäne
 Kundenverwaltung 567
 Anwendungshierarchie 78
 Suche 130
 Anwendungskomponente 125, 126
 Hierarchie 125
 Hierarchiesuche 130
 Modellierungskonstrukt 126
 Paket 127
 Überblick 126
 Anwendungsleiste 61
 Anwendungsserver 53
 ANY 186, 190
 Anzahl der Datensätze 607
 APPEND 191
 Applikationsschicht 40, 133
 Applikationsserver 30, 133
 Architektur 41
 DATASET 681
 Datei erreichbar 681
 Daten 666, 681
 Datenablage 681
 Applikationswissen 123
 Arbeitsbereich
 DB-Tabelle 646
 strukturieren 646
 Arbeitsprozess 134, 565
 Datenbankschnittstelle 565
 Datenbankverbindung 135
 Ressource 135
 Arbeitsstruktur 191
 befüllen 191, 193
 Architektur
 Aufgabe 132
 Röntgenblick 132
 Schicht 132
 ARITHMETIC_ERRORS 414
 Arithmetische Operation 211
 Array-Fetch 611
 Beispiel 630
 AS 633
 ASCENDING 192
 ASCII 159
 ASCII-Format 676
 IBM-Codepage 676
 AS ICON 478
 Assembler 39, 42
 Assembler-Makros 42
 ASSIGN 689, 691, 695, 697
 ASSIGN COMPONENT
 dynamisch 698
 ASSIGNING 690
 Assoziation 280, 293, 317, 341
 implementieren 326
 Kardinalität 280
 AS SYMBOL 478
 Asterisk 503, 549
 AT 631
 AT LINE-SELECTION 109, 233, 484
 Atomicity 637
 AT SELECTION-SCREEN 233,
 468, 470
 lokale Daten 233
 AT SELECTION-SCREEN ON 475
 AT SELECTION-SCREEN ON
 BLOCK 475
 AT SELECTION-SCREEN ON
 HELP-REQUEST 476
 AT SELECTION-SCREEN ON
 VALUE-REQUEST 476
 AT SELECTION-SCREEN
 OUTPUT 233, 464, 470
 Attribut 279, 292
 CLASS-DATA 297
 DATA 297
 Instanz 295, 297
 Karteireiter 322
 Klasse 297
 Read-Only 322
 statisches 296, 313
 zugreifen 310
 Attributdefinition 292
 Grundform 292
 Attributpflege 322
 AT USER-COMMAND 233
 Aufgabe 73
 Entwicklungsobjekt zuordnen 73
 freigeben 74
 Aufgabendomäne 270
 Aufgabenstellung 339
 Aufrufkaskade
 Methode 428
 Aufsteigend sortiert 611
 Auftrag
 abgeschlossener 73
 Aufgabe 73
 Bearbeiter 76
 Bezeichnung 76
 freigeben 77
 geschützter 73
 Mitarbeiter zuordnen 73
 temporärer 77
 Auftraggeber 519
 Aufwärtskompatibilität 96
 Ausführbares Programm 54
 Zeitpunkt 229
 Ausführen 477
 Programm 43
 Ausgabe
 Liste 200
 Auslieferungsklasse 568
 Ausnahme 411
 auslösen 403, 410, 421, 422
 Behandelbare 403
 behandeln 409, 411, 424, 671
 Behandlungsentscheidung 424
 Behandlungsmuster 409, 412
 CATCH SYSTEM-EXCEPTIONS 406
 definieren 410
 Drucktaste 423
 Einstiegsbild 419
 END SYSTEM-EXCEPTIONS 406
 Explizite 403
 Feigling 411
 get_text 420

- Globale Klasse 423
 - Implizite 403
 - Java 417
 - Nachricht 421
 - Nachrichtenklasse 420
 - Nicht behandelbare 403
 - Objektorientierte 417
 - OTHERS 407, 414
 - Werfen 407
 - Zwischenklasse 419
 - Ausnahmebehandlung
 - Strategie 409
 - Ausnahme-Catcher 416
 - Ausnahmegruppe 407
 - Ausnahmeklasse 321, 407, 417
 - Aktivieren 420
 - cx_dynamic_check* 417
 - cx_no_check* 417
 - cx_root* 417
 - cx_static_check* 417
 - Hierarchie 417
 - Karteireiter Texte 420
 - Klassenhierarchie 420
 - RAISE EXCEPTION 422
 - RAISING 421
 - UML 417
 - Ausnahmekonzept 404
 - Ausnahmengespicktes Projekt 425
 - Ausnahmenstaubsauger 412
 - Ausnahmeobjekt 417, 420
 - fangen 424
 - Ausnahmesituation
 - Reaktion 409
 - Ausnahmetext 424
 - Ausschlagen 368
 - Ausstellungszentrum 29
 - Austauschdaten 681
 - Auswahl
 - 1 aus n 443
 - m aus n 442
 - Auswahlkalender 440
 - Auswahlliste 521
 - Auswertungen 42
 - Auswirkung
 - lokale Variable 246
 - AUTHORITY-CHECK 550
 - Eingabefeld 556
 - Muster 556
 - AUTHORITY_CHECK_DATASET 682
 - Autorisiert 548
- ## B
- Babuschka 127
 - Backtick 239
 - BAPI_TRANSACTION_COMMIT 639
 - BAPI_USER_GET_DETAIL 252
 - Basis 40
 - Basisdienste 43
 - Basisklasse 342
 - Basistabelle 620
 - View 615
 - Batch 39, 42
 - Batch-Job 39
 - Bauplan
 - Klassendiagramm 281
 - Stuhl 161
 - BCD_ZERODIVIDE 414
 - Bearbeitungsmodus 146
 - ändern 146
 - Bedingung 214
 - prüfen 611
 - Beet 433
 - Visualisierung 481
 - Befehlsfeld 57, 539
 - BEGIN OF 176, 293
 - BEGIN OF BLOCK 460
 - Beglückung
 - Maximale 439
 - Behälter → Container
 - Behandlung
 - Implementierungsmuster 424
 - Benutzeraktion 434
 - Benutzerdefiniert lokal 168
 - Benutzerinteraktion 646
 - Benutzerkommando 473
 - Benutzermenü 63
 - Benutzeroberfläche 490, 502, 508
 - Drucktastenleiste 508
 - Element 490
 - Funktion 508
 - Funktionstastenzuordnung 508
 - Menüleiste 508
 - Setzen 504
 - Status 509
 - Symbolleiste 508
 - Titelleiste 508
 - Benutzerpflege 500
 - Benutzersitzung 135, 448
 - Benutzerspezifische Einstellung 140
 - Benutzerstamm 550
 - Benutzerunterstützung 458
 - Benutzerverteilung 53
 - Berechnung
 - Ergebnis 525
 - Kilokalorien 536
 - Berechtigung 547, 548, 549
 - alle 549
 - Anwender 559
 - Benutzerstamm 556
 - Berechtigungsobjekt 549
 - DATASET 682
 - ermitteln 559
 - Wert 549
 - zugeordnete 549
 - Berechtigungen 30
 - Berechtigungsadministrator 550
 - Berechtigungsfeld 550
 - falsches 558
 - Berechtigungskonzept 599
 - Berechtigungsobjekt 548, 551
 - anzeigen 551
 - Attribut 548
 - Aufbau 554
 - Berechtigung 549
 - Beschreibung 555
 - Dokumentation 555
 - Feld 548
 - Klasse 548, 549
 - Name 555
 - Objektklasse 552, 555
 - S_TCODE 548
 - Suche 552
 - wiederverwenden 551
 - Berechtigungsprojekt 555

- Berechtigungsprüfung 232, 549
 - AUTHORITY-CHECK 550
 - DUMMY 558
 - erfolgreiche 550
 - Ergebnis 559
 - fehlgeschlagene 558
 - implementieren 556
 - Berechtigungsverwaltung 549
 - Bereich
 - reservieren 114
 - Beschleunigung 688
 - Beschreibung 161
 - Beschreibungsobjekt 366, 467, 694
 - Beständige Daten 182
 - Bestätigung
 - überschreiben 676
 - Besucherkzentrum 29
 - Betriebssystem
 - Schnittstelle 43
 - BETWEEN 215
 - Beziehung
 - 1 zu n 567
 - Bibliothek 193
 - Bier 338
 - bestellen 364
 - obergäriges 341
 - untergäriges 341
 - Big-Oh-Notation 186
 - Bild
 - darstellen 114
 - Bildchen 669
 - Bildobjekt 118
 - Bildschirm 33
 - Binärformat 676
 - Binärstring 159
 - Bindestrich 102, 177
 - Birchermüsli 454
 - Bisschen verrückt
 - Schwaiger Roland 503
 - Bit 159
 - Blaupause 161
 - Block 148
 - Blume 478
 - Icon 467
 - Bohrendilemma 334
 - Bohnenklasse 321
 - Boolescher Typ 201
 - Bottom-up 238
 - BOUND 118
 - Braumeister 341
 - Braune Spalte 152
 - Breakpoint
 - Symbol 152
 - unbedingter Haltepunkt 150
 - Brille 141
 - Browser 139
 - Auswahl 140
 - einstellen 140
 - Browser schließen 139
 - Browser-Sicht 142
 - Bücherregal 183
 - Bündeln
 - Änderung 643
 - Business ByDesign 41
 - Business Functionality 41
 - Business-Komponente 125
 - Business Object Layer 397
 - Business-Objekt 125, 126
 - Business Suite 7 41
 - Button
 - Icon 669
 - BYPASSING BUFFER 605
 - Byte 159
 - Oktett 159
 - Bytecode 45
 - Bytefolge
 - XSTRLEN 205
- ## C
- Call-By-Reference 240, 241, 304, 323
 - Call-By-Value 240, 304, 323
 - Call-By-Value-And-Result 241
 - CALL FUNCTION
 - dynamisch 700
 - CALL METHOD 118, 314
 - dynamisch 700
 - CALL SCREEN 109, 133, 492, 533
 - CALL SELECTION-SCREEN 435
 - Cannabaceae 452
 - Captain Blackbeard 294
 - CASE 102
 - ENDCASE 216
 - Fallunterscheidung 216
 - Verwendung 216
 - WHEN 216
 - WHEN OTHERS 216
 - CASE WHEN ENDCASE 216
 - Cast
 - Ausnahmebehandlung 371
 - Down 369
 - Up 365
 - CASTING 692
 - Casting-Operator 369
 - CASTING TYPE 697
 - CATCH 371, 424
 - INTO 424
 - CATCH SYSTEM-EXCEPTIONS 406, 414
 - Aufrufhierarchie 415
 - OTHERS 414
 - Schachtelbar 415
 - CHANGING 240, 299, 305, 534
 - Aufruf 247
 - praktisch 242
 - value 241
 - Character 166
 - Checkbox 443, 482
 - CHECKBOX 441
 - cl_abap_elemdescr 467
 - cl_abap_elemdescrs 366
 - cl_abap_structdescr 366, 692
 - cl_abap_typedescr 366, 692
 - describe_by_name 366
 - describe_by_object_ref 366
 - CLASS 284
 - DEFINITION 284
 - IMPLEMENTATION 285
 - Class Builder 320
 - Einstiegsmaske 320
 - class_constructor 308
 - CLASS-DATA 297, 382
 - CLASS-METHODS 300, 382
 - Class Responsibility
 - Collaboration 273
 - CLEAR 204, 659

cl_gui_alv_grid 349
cl_gui_cfw
 flush 671
cl_gui_custom_container 109, 349
cl_gui_frontend_services 667
 Abfragefunktion 667
 clipboard_export 667, 674
 Dateifunktion 667
 Environment 667
 get_temp_directory 667, 671
 gui_download 667, 676
 gui_upload 667
 Standarddialog 667
cl_gui_picture 109
cl_gui_timer 219
cl_http_client 427
 create 427
cl_http_utility
 set_request_uri 428
Client 55
 Daten 666
 instanzieren 427
 SAP-System 425
Client-Server-Architektur 40, 55
CLIENT SPECIFIED 605
Clipboard 674
 lesen 667
 schreiben 667, 673, 674
clipboard_export
 FLUSH 675
CLOSE DATASET 681, 683, 685
Cloud 31
cl_salv_table 695
Cluster-Tabelle 594
CO 213
Code
 Produktion 263
Code einfügen
 Berechtigungsprüfung 556
Code Reuse 228
Code-Snippet 442
Codevorlage 149
Codex Hammurapi 456
Codierung 161, 681
COLOR 480

COMMIT 638
 explizites 639
 implizites 639
 WORK 639
Compiler 45
COMPUTE 42
Computer
 hobeln 161
CONCATENATE 205, 208
 Einschränkung 205
 INTO 206
 SEPARATED BY 206
 Syntax 206
CONDENSE 206, 208
 Leerzeichen entfernen 206
 string 206
Consistency 637
CONSTANTS 298, 335
Constructor 307
 Definition 307
 Initialisierung 307
Container 77, 118, 255, 326
CONTINUE 220
 verwenden 226
Control Framework 118, 349, 671
 Klasse 671
Controlling 40
Copy-&-Paste-Programmierer 228
CORRESPONDING FIELDS OF 611
CP 213
CRC 273
 Class 273
 Collaboration 273
 Kärtchen 273
 Responsibility 273
 UML 279
Create 636
CREATE DATA 691, 694
 dynamisch 697
CREATE OBJECT 118, 289
 Abstraktes 374
 Constructor 307
CREATE PRIVATE 398
CRUD 599, 636
Custom Control 114

Custom Control Area 118
 Name 115
cx_root 417, 424
CX_SY_ 419

D

DATA 105, 160, 292, 297, 382
 lokal 245
Data Encapsulation 286
DATASET 681
 Berechtigung 682
 READ 685
Datei
 Inhalt 684
 lesen 681
 Name 684
 öffnen 681, 682
 Speicherort 681
 Status 684
 technische Information 684
 Textmodus 685
Dateiende 685
Dateimanager 511
Dateiname 669, 682
 betriebssystemabhängiger 682
Dateioperation 666
Dateipfad 669
Dateischreiben 676
Dateisystem 666
Datei-Up/Download 667
Dateiverzeichnis
 Applikationsserver 683
Daten 30
 abhängige 642
 ändern 655
 Aufteilung 578
 Datei 666
 entfernen 636
 lesen 595
 performant einfügen 654
 persistieren 562
 puffern 597
 schreiben 636
 schützen 548
 speichern 158, 561

- Daten (Forts.)
 - verarbeiten* 158
 - verknüpfen* 614
 - View* 615
 - Datenänderung
 - Massendaten* 662
 - protokollieren* 577
 - Datenart 575
 - Datenbank 562, 596
 - Kommunikation* 135, 596
 - logische* 434
 - Normalisierung* 579
 - SAP* 563
 - unabhängige* 564
 - zusätzliches Feature* 596
 - Datenbankabhängige Schicht 566
 - Datenbanklast 664
 - Datenbank-LUW 638
 - Logical Unit of Work* 637
 - Datenbankmanagementsystem 596
 - Datenbankobjekt 591
 - anzeigen* 591
 - Element* 591
 - Primärindex* 591
 - Sekundärindex* 591
 - Datenbankschicht 40
 - Datenbankschnittstelle 564, 597
 - Datenbanksperre 640, 663
 - DB-LUW* 640
 - Datenbanksprache 596
 - Datenbanksystem 83
 - ohne* 665
 - Datenbanktabelle 71, 83, 567, 595
 - administrative Einstellung* 93
 - aktivieren* 94
 - ändern* 661
 - anlegen* 91, 568, 598
 - Auslieferungsklasse* 92
 - Auslieferung und Pflege* 92
 - Beschreibung* 91, 564
 - Data Browser/Tabellensicht-Pflege* 92
 - Datenart* 93
 - Daten einfügen* 636, 651
 - Datenlöschung* 658
 - Design* 83
 - einfügen* 650
 - Einträge erfassen* 94
 - Eintrag erzeugen* 592
 - Erweiterungskategorie* 93
 - erzeugen* 590, 591
 - Feld* 92
 - Größenkategorie* 93
 - lesen* 100, 224
 - Massendatenänderung* 656
 - Name* 91, 565
 - physische* 564
 - Primärschlüssel* 92
 - schreiben* 650
 - Spalte* 83, 565
 - Spalte definieren* 570
 - technische Einstellung* 93
 - Typ* 199
 - überschreiben* 655
 - Datenbanktabelle
 - Zugriff beschleunigen* 688
 - Datenbankverbindung 135
 - Datenbank-View 615
 - Definition* 618
 - inner Join* 616
 - Datenbankzugriff 96, 597
 - Datenbereich
 - reservieren* 692
 - Datenbeschaffung 476
 - Datenbestand
 - bearbeiten* 596
 - LOOP* 192
 - Datencontainer 563
 - Datendeklaration 287
 - Objektreferenz* 287
 - Dateneingabe-Modus 62
 - Datenelement 84, 163, 170, 570
 - aktivieren* 90, 172
 - anlegen* 89, 170, 572
 - Datentyp* 171
 - Detailpflege* 171
 - Domäne* 89, 173
 - Domäneninfo befüllt* 174
 - eingebauter Typ* 171
 - Feldbezeichner* 90
 - Kurzbeschreibung* 89, 171
 - Name* 89, 170
 - semantische Ebene* 85
 - semantischer Typ* 84
 - Typ* 85
 - vordefiniertes* 179
- Datenformat
 - Download* 676
 - Datenhaltung 562
 - Datenkapselung 286
 - Datenkodierung 682
 - Datenkommunikation 109
 - Datenmanipulationsoperation 645
 - Datenmodellierung 567
 - Datenobjekt 158, 159
 - definieren* 160
 - interner Modus* 159
 - kompatibles* 203
 - Name* 160
 - zuweisen* 203
 - Datenorientierung 269
 - Datenpufferung 182
 - Datenredundanz 579
 - Datenreferenz 691
 - Datenrefs 690
 - eindeutig finden* 195
 - puffern* 611
 - schreiben* 593
 - Datensatz 83, 563
 - eindeutig finden* 195
 - puffern* 611
 - schreiben* 593
 - Datenschutzbereich 663
 - Datenselektion 434
 - Datensicht 614
 - Datenspeicherung 158
 - DatenTRANSFER 683
 - Datentransport 527, 529, 530
 - automatischer* 112, 534
 - Namensgleicher* 529
 - Datentyp
 - Auswahl* 170
 - Domäne* 173
 - Plan* 161
 - Datenübertragung
 - Codierung* 681
 - Datenverarbeitung 100
 - Datenzustand
 - konsistenter* 638

Datepicker 440, 458
 dats 440
 Datum
 formatieren 239
 DB-Administrator 586
 DB_COMMIT 639
 DB-LUW 638
 Beginn 640
 SAP-LUW 642
 Sperre freigeben 640
 DB-Programmierung
 fünf Gebote 664
 DB-Schnittstelle 566
 DB-Tabelle 564
 anlegen 564
 Debugger 122, 150, 152
 Abarbeitungsposition 153
 Detailanzeige 155
 Einzelsschritt 153
 Einzelsschrittnavigation 152
 /i 150
 Inhaltsanzeige 153
 Inhaltsanzeige Variable 152
 klassischer 155
 Select-Option 451
 starten 150
 Debugger-Start
 Breakpoint 150
 DECIMALS 167
 DEFAULT 439
 Defect 228
 DEFINITION 284
 Definitionsdetail
 Berechtigungsobjekt 554
 Definitionsfenster
 Berechtigungsobjekt 554
 Definitionsstelle
 Typ 360
 Definitionsteil 284
 Deklaration
 Selektionsbild 436
 Deklaratives Schlüsselwort 97, 160
 Dekomentieren 48
 Delete 636
 DELETE 196, 597, 636, 658
 Arbeitsbereich 658
 dynamisch 697
 FROM 658
 FROM TABLE 659
 Indexvariante 196
 Primärschlüssel 658, 660
 Schlüsselangabe 196
 sy-dbcnt 658, 660
 sy-subrc 658, 659, 660
 WHERE 658, 659
 DEQUEUE 663
 DESCENDING 192
 DESCRIBE 208
 Design Pattern 397
 Desktop-Client-UI 40
 Detailanzeige 155
 Detailpflege
 Domäne 174
 DIAG-Protokoll 462, 498
 Dialogprogrammen 42
 Dialogschritt
 SAP-LUW 642
 Dialogschritt abgeschlossen
 Commit 639
 Dialogtransaktion 539
 Dictionary 567
 globaler Typ 163
 Dictionary-Objekt
 Kontextmenü 91
 Dictionary-Referenz 611
 Dictionary-Struktur
 fertige 181
 SYST 530
 Dict-/Programmfelder-Fenster 112
 Direkte Adressierung 288
 direkter Zugriff 688
 DIR_TEMP 683
 Dispatcher 133
 Dispatcher Queue 134
 disp+work 43
 DIV 211
 DIVIDE 211
 Division 211
 DML 598
 Dobos-Torte 456
 DO ENDDO 217
 Dokumentation 100, 171, 405
 Systemfeld 405
 Dokumentenklasse 58
 Dokumentenpflege 58
 aktivieren 59
 Änderungsmodus 59
 Dokumentenklasse 58
 Editor 59
 Name 58
 sichern 59
 Sprache 58
 Domäne 84, 85, 173
 aktivieren 88, 174
 anlegen 86, 174
 Datenelement 174
 Datentyp 174
 Detail 87
 Kleinbuchstabenrelevanz 87
 Kurzbeschreibung 174
 Name 87
 Paketzuordnung 88
 technische Eigenschaft 174
 technischer Typ 84, 85
 Wertebereich 582
 Wertetabelle 582
 wiederverwenden 173
 Zusatzangabe 85
 Domänenfestwert 467
 Domänenwissen 270
 Don Quijote 295
 Doppelklick
 Navigation 144, 497
 Verzweigungsliste 484
 Doppelpunkt 47
 Double-Binding-Problem 138
 Doughnut 353
 Down-Cast 369
 Nicht eindeutig 369
 Download 666, 673
 Daten 676
 SAP-System 31
 Sicherheitsabfrage 679
 Ziel 676

- Drag & Drop 238
 - Unterprogramm-Aufruf* 247
- Drei Amigos 276
- Dreistelliges Kürzel 462
- Drei-System-Landschaft 73
- Drucktaste 434, 490, 514, 645
 - 35 Stück 514
 - anlegen* 514
 - auswerten* 648
 - Bezeichnung* 647
 - Funktionscode* 645, 647
 - implementieren* 646
 - positionierte* 647
 - Selektionsbild* 647
 - Text* 647
- Drucktastenbehandlung 670
- Drucktastenleiste 61, 508, 514
 - Aktivieren* 146
 - Anzeigen <-> Ändern* 146
 - Gestaltung* 646
 - Position* 515
 - Prüfen* 146
 - Testen/Ausführen* 146
- DUMI 636
- DUMMY 558
- Durability 637
- Durchlauf
 - sy-index* 217
- Dynamic Information and Action
 - Gateway* 498
- Dynamische Analyse 150
- Dynamische Bedingung 698
- Dynamische Bedingungs-
spezifikation 696
- Dynamische Feldspezifikation 696
- Dynamische Komponenten-
spezifikation 696
- dynamischen Programmierung
 - Fehlerprüfung* 696
- Dynamischer Aspekt 316
- Dynamischer Typ 165, 364, 365, 697
 - Bestellung* 364
- Dynamisches Feld 697
- Dynamische Sicht 316
- Dynamisches Programm 236
 - Dynamisches Token 692, 695, 696
 - Dynamisches Unterprogramm 700
 - Dynamische Typisierung 166
 - Dynamische Typspezifikation 696
 - Dynamische Unterprogramme 696
 - Dynamisierung
 - Motivation* 688
 - Dynpro 71, 108, 236, 487, 490
 - Ablauflogik* 108, 229, 236, 487, 488, 495, 497
 - aktivieren* 118
 - anlegen* 110
 - Aufruf* 492
 - aufrufen* 108, 109
 - automatische Codeproduktion* 504
 - Client-Seite* 498
 - Darstellung* 487
 - Daten transportieren* 108
 - DIAG-Protokoll* 498
 - Dictionary-Struktur* 520
 - dynamisches Programm* 108
 - Dynpro-ABAP* 502
 - Eigenschaft* 487, 495
 - Einstellung* 495
 - Elementliste* 495, 527
 - Empfehlung* 520
 - Enter* 501
 - Ereignisblock* 229
 - Folge-Dynpro* 111, 495, 496, 500
 - Folge-Dynpro-Rezept* 501
 - gestalten* 108, 112, 519
 - Include* 503
 - INPUT-Modul* 504
 - Kontextmenü* 495
 - Kurzbeschreibung* 495, 496
 - Layout* 112, 488, 498, 522
 - Modales Dialogfenster* 496
 - Modul* 116, 502
 - Nummer* 497
 - Oberfläche* 508
 - OUTPUT-Modul* 504
 - PAI* 116
 - PBO* 116
 - Pflege* 494
 - Programmblock* 502
 - Programmierung* 488
 - Rahmenprogramm* 503
 - Rufen* 539
 - Serverseite* 498
 - Subscreen* 496
 - technische Einstellung* 108
 - Transaktion* 492
 - Transaktionscodefeld* 499
 - Typ* 495, 496
 - Unterprogramm* 534
 - Visualisierung* 236
 - Dynpro-ABAP 117, 236, 502
 - Dynpro-Bereich 62
 - Dynpro-Programmierung 236
 - Dynpros 42

E

 - Edgar F. Codd 563
 - Editor
 - Version* 147
 - Editorzeile 96
 - Editorzeilen 48
 - EDV 159
 - Eier
 - suchen* 206
 - Eigenartige Syntax 386
 - eigener Prozess
 - Kern* 43
 - Eigenschaft 487
 - Eigenschaftenfenster 526
 - Eindeutigkeit 190
 - Eingabe 490
 - Eingabebereit 526
 - Eingabebereites Feld 475
 - Eingabedefizit 458
 - Eingabeelement 104
 - Eingabefeld 97, 104, 490
 - anpassen* 114
 - Label* 105
 - Text* 104
 - Eingabehilfe 236, 439, 458, 476, 524, 611
 - automatische* 458
 - Festwert* 458

- Prüftabelle 458
- Suchhilfe 458
- Eingabehilfe/-prüfung 584, 586
- Eingabeintervall 452
- Eingabekombination 475
- Eingabeunterstützung 440
- Eingabe-Verarbeitung-Ausgabe 158
- Eingebauter ABAP-Typ 168
 - f 163
 - i 163
 - Tabelle 168
- Einheit 167
- Einkaufsliste 477
- Einrückung 149
- Einsicht
 - Anforderung 339
- Einsprungsmarke 464
- Einstellung 64, 140
 - ABAP Editor 141
 - Workbench allgemein 140
- Einstellungsfenster 140, 148
- Einstiegs-Dynpro 495, 521
- Eintrag
 - erfassen 592
 - pflegen 592
- Einzelfeld 604
 - Ausgabe 200
 - Zugriff 200
- Einzelatz 601
 - einfügen 650
 - lesen 195
- Einzelatz-Lesen 200
- Einzelatzoperation 195
- Einzelschritt 153, 155
- Einzelstrittnavigation 152
- Einzelwert
 - aufnehmen 129
 - ausschließen 129, 454
 - selektieren 129, 453
- Eiweißrast 343
- elegante Zugriffe
 - Feldsymbol 688
- Elektronische Datenverarbeitung 159
- Elementarer Typ 163
 - globaler 170
- Elementliste 527, 528
- ELSE 102, 214
- ELSEIF 214
- Empfänger 330, 332
- ENCODING DEFAULT 682
- ENDAT 631
- ENDCASE 103, 216
- ENDCATCH 414
- ENDCLASS 285
- ENDFORM 239, 299
- ENDIF 102
- ENDINTERFACE 381
- ENDLOOP 192
- Endlosschleife 220
- ENDMETHOD 300
- ENDMODULE 237, 504
- END OF 176, 293
- END-OF-PAGE 233
- END-OF-SELECTION 233, 465, 477, 493
- ENDSELECT 602, 612
- END SYSTEM-EXCEPTIONS 406
- ENDTRY 424
- ENDWHILE 221
- Enjoy-Control 108, 109
 - CREATE OBJECT 118
- ENQUEUE 663
- Enter 501
- Enterprise Resource Planning 41, 599
- Enterprise Software 38
- Entitätstyp 567, 570
- Entschleunigung 688
- Entwicklung
 - Organisation 122
 - Werkzeug 137
- Entwicklungsklasse 77
- Entwicklungsobjekt 73
 - bearbeiten 72
 - hierarchisches 142
 - Kategorie 142
 - Kontextmenü 143
 - Teilobjekt 142
 - temporäre Zuordnung 77
- Entwicklungsobjektkategorie 146
- Entwicklungsobjektliste 139
- Entwicklungsprojekt 73
- Entwicklungsumgebung 43
- Entwicklungswerkzeug 144
- Entwurfsmuster 397
- EQ 213, 611
- Erben 338, 341
- Erbt von 359
- Erdbeertorte 298
- Ereignis 330, 331, 464
 - AT LINE-SELECTION 484
 - AT SELECTION-SCREEN OUTPUT 464
 - Behandlermethode 332
 - Behandlerregistrierung 333
 - END-OF-SELECTION 465
 - INITIALIZATION 465
 - LOAD-OF-PROGRAM 464
 - PBO 236
 - START-OF-SELECTION 229, 465
 - TOP-OF-PAGE 479
- Ereignisbehandler-Methode 332
- Ereignisblock 229
 - globale Daten 233
 - keine Schnittstelle 233
 - LOAD-OF-PROGRAM 230
 - Reihenfolge 230
 - START-OF-SELECTION 230
 - Zeitpunkt 230
- Ereignisschlüsselwort 97, 230, 464
 - einleitendes 230
 - Reihenfolge 464
- Ereigniszeitpunkt
 - Ende 465
- Erfolgsgeschichte 40
- Ergebnismenge
 - inner Join 626
- Erich Gamma 397
- ERP 599
- Erweiterungskategorie 93, 180, 521, 589
 - beliebig erweiterbare 93, 181, 590
 - erweiterbare 181
 - Menüeintrag 589
 - nicht erweiterbare 181
 - nicht klassifizierte 181

- Erweiterungskategorie (Forts.)
 - Pflege* 181
 - Pflegemöglichkeit* 181
 - Struktur* 180
 - Erzeugung
 - explizite* 490
 - implizite* 490
 - escapen 699
 - Escape-Symbol 129, 699
 - EVA 158
 - Event 330
 - Exception Builder 420
 - EXCEPTIONS 305, 406, 410
 - Kommentieren* 412
 - Exemplar 161
 - EXIT 217, 220, 427, 474
 - exotische Programmierung 691
 - Explizite Erzeugung 490
 - expliziter Typ 183
 - Explizite Typisierung 183
 - Export
 - zu Import* 259
 - EXPORTING 305
 - Exportparameter
 - definieren* 261
 - Externer Modus 135
 - Extrakte 42
- ## F
- F1 414
 - Fachwissen 270
 - Falle
 - Namenslänge* 438
 - Fallunterscheidung 216
 - Farbe 148
 - POSITIVE* 482
 - Favorit 65
 - Datei* 65
 - Transaktion* 65
 - Webadresse* 65
 - Fehlerdialog 639
 - Commit* 639
 - Fehlermeldung
 - Deklaration* 166
 - Fehlersuche 228
 - Fehlerteufel 657
 - Feld 102, 158
 - Ausgabe* 199
 - Berechtigungsobjekt* 548
 - Eingabebereites* 475
 - initialisieren* 204
 - Karteireiter* 570
 - nicht identifizierendes* 585
 - platzieren* 114
 - Wert* 549
 - Feldbezeichner 90, 171, 172, 463
 - Feldsymbol 688
 - Beispiel* 689
 - Feldsymboldeklaration 690
 - Fenster
 - externer Modus* 136
 - Festplattenplatz 33
 - FIELD-SYMBOLS 689
 - FILE_GET_NAME 682
 - Filter
 - View* 627
 - Final 321
 - Finale Klasse 375
 - Finanzbuchhaltung 38
 - Finanzen 40
 - FIND 206, 208
 - erstes Vorkommen* 206
 - Groß- und Kleinschreibung* 206
 - Zusatz* 206
 - Fingerübung 193
 - Flache Struktur 181
 - Fließkommazahl 163
 - Float 163
 - Flüchtige Daten 182
 - Fluchtsymbol 129
 - Folge-Dynpro 111, 495, 496
 - Dynamisches* 496
 - Statisches* 496
 - Folgesystem 142
 - FOR 451
 - FOR INPUT 685
 - FORM 239, 299
 - Formalparameter 240, 299, 304
 - links* 263
 - typisieren* 243
 - Unterprogramm* 247
 - Format 666
 - FORMAT 480
 - Form-Routine 239
 - FOR OUTPUT 681
 - Fortgeschrittenes Thema 217
 - Fragestellung
 - wozu* 339
 - Fremdschlüssel 578
 - anlegen* 581
 - Kardinalität* 583, 585
 - Symbol* 581
 - View* 617
 - Fremdschlüsseldefinition 582, 583
 - Ergebnis* 584
 - Pflegedialog* 581
 - übernehmen* 584
 - Vorschlag* 592
 - Fremdschlüsselfeld 581, 583
 - Art* 583
 - Fremdschlüsseltabelle 579
 - Fremdschlüsselverprobung 581, 584, 592
 - Beispiel* 611
 - Eingabeprüfung* 646
 - FROM source 605
 - Frontend Editor (neu) 147
 - Frontend Service 668
 - F-Tastenzuordnung 62
 - Fuchsschwanzgewächs 452
 - Function Builder 252, 411
 - Einstiegsbild* 252
 - Karteireiter Eigenschaften* 258
 - Funktion 54, 205, 211, 508, 512
 - aktive* 508
 - Aufruf* 205
 - Eigenschaft* 512
 - Funktionstext* 513
 - Ikonenname* 513
 - Ikonentext* 513
 - inaktive* 508
 - Infotext* 513
 - Leerzeichen* 205
 - Text* 513
 - Wiederverwendung* 228, 516
 - Zielvariable* 205
 - Funktionale Methode 305

- Funktionsbaustein 252
 - Aufruf 254, 263, 314
 - Aufruf Drag & Drop 263
 - Aufruf erzeugen 411
 - Aufruf manuell 263
 - Aufrufmuster 263
 - Ausnahme 259, 406
 - Behandlungsmuster 412
 - Changing 259
 - Daten 253
 - Datenspeicherzugriff 264
 - definieren 254
 - Eigenschaft 255, 258
 - entwickeln 260
 - EXCEPTIONS 406
 - Export 259
 - EXPORTING 240, 253
 - Formalparameter 263
 - ICON_CREATE 410
 - Implementierung 260
 - Import 259
 - IMPORTING 240
 - Importing-Abschnitt 253
 - Importparameter 253
 - Laufzeit 253
 - lokale Daten 254
 - Objektliste anzeigen 255
 - OTHERS 412
 - Schnittstelle 254, 304, 406, 412
 - Schnittstellendefinition 258
 - Tabelle 259
 - Teilobjekt 262
 - testen 252
 - Testen/Ausführen 411
 - Testrahmen 411
 - Unterprogramm 252
 - Funktionscode 512, 517, 645, 648
 - Aktiver 517
 - Drucktaste 647
 - Inaktiver 517
 - Variable 530
 - Funktionsgruppe 254, 302
 - anlegen 257
 - Container 255
 - Daten 255
 - Datenspeicher 264
 - Detail 257
 - Dynpro 255
 - Ereignis 255
 - Funktionsbaustein 254
 - globale Daten 254
 - Include 255, 256
 - interner Modus 264
 - Kontextmenü 260
 - Name 255
 - Programm 255
 - SU_USER 256
 - Unterprogramm 254, 255
 - Funktionstaste 517, 518
 - Zuordnung 61, 62, 508, 515, 518
 - Funktionsumfang 42
 - Fußgesteuerte Schleife 220
- ## G
- Gang of Four 397
 - Ganzzahlige Division 211
 - Garbage Collector 329
 - Gartendatum 433
 - GE 611
 - Gebackene Maus 380
 - Geisterbild 113, 523
 - GENERATE SUBROUTINE POOL 700
 - Generischer Typ 293
 - Gepackte Zahl 167
 - Byte 167
 - Geschachtelte Struktur 181
 - Geschäftsfunktionalität 41
 - Geschäftsobjekt 126
 - Geschäftsprozess 123
 - Geschütztes Attribut 352
 - GET 428
 - GET CURSOR 485
 - GET-Parameter 447
 - Getter 279
 - Gewöhnliche ABAP-Klasse 321
 - Gleichheitsbeziehung 617
 - Globale Klasse 320
 - Interface-Attribut und Interface-Methode 396
 - Interface implementieren 395
 - Interface zuordnen 396
 - Globaler elementarer Typ 168, 170
 - Globaler strukturierter Typ 178
 - Globaler Tabellentyp 189
 - verwenden 191
 - Globaler Typ 87, 163
 - anlegen 170
 - Globales Interface 393
 - Globale Struktur
 - Definition 178
 - Globale Variable
 - problematische 246
 - Goldenes Objekt erster Klasse 336
 - GOTO 464
 - Grady Booch 276
 - Grafische Modellierung 276
 - Grafischer Layout-Editor 522
 - Graphical Screen Painter 112
 - Größenkategorie 576
 - Größer 611
 - Größer als 454
 - Größer gleich 611
 - Groß-/Kleinkonvertierung 141
 - Groß-O-Notation 186
 - Groß- und Kleinkonvertierung 149
 - GROUP BY 608
 - Grundliste 484
 - Grundrechenart 211
 - Grüner Pfeil 518
 - Gruppe
 - logische 475
 - Gruppenstufe 631
 - AT 631
 - ENDAT 631
 - Gruppenstufenverarbeitung 630
 - Beispiel 630
 - Gruppierung 460
 - GT 611
 - GUI
 - Anweisung 133
 - gui_download
 - Formalparameter 676
 - GUI-Fähigkeit 540
 - GUI-Klasse 118

GUI-Programmierung 118
gui_upload 676
Gültiger Wert
 Prüfung 646

H

Halbbyte 167
Hammer und Schraubenschlüssel 446
Hänsel-und-Gretel-Sanduhr 217
HASHED 186
 einfügen 192
 Feldsymbol 690
 INSERT 192
 Schlüssel 187
Hash-Key 185, 186
Hash-Tabelle 185
Hauptmodus 135
Hauptpaket 77, 127
Hauptspeicher 33, 158, 166
 adressieren 158
HAVING 608, 664
Help-View 615
 outer Join 616
Henne-Ei-Problem 158, 290
HIDE 484
Hide-Bereich 484
HIGH 450
High Performance Analytic
 Appliance 580
Hilfe 171, 476, 512
 Ausnahmegruppe 414
 F1 414, 476
 F4 476
Hilfsmittel 140
Hinterglas-Schrödinger 172
Hintergrund 435
Holen aus Dictionary 113
Holzboot 368
HOME 78
Hopfen kochen 363
Hostname 33
HOTSPOT 482, 485
HTTP 425, 428
 GET 428

Nachrichtenkopf 428
 Nachrichtenkörper 428
 Version 428
HTTP-Client 427
HTTP-Port 427
HTTP-Request 425
Humulone 363
Humulus lupulus 452
Hypertext Transfer Protocol 428

I

IBM DOS 39
Icon 513
 Konstante 669
ICON 411, 478
ICON_CREATE 410, 412, 669
Icon mit Text 669
ICON_SELECTION 478
IDE 72
IF 214
 logische Bedingung 215
IF-Anweisung 214
if_http_client 427
IF THEN ELSE 214
Ikonenname 513
Ikonentext 513
Implementationsteil 284
Implizite Erzeugung 490
impliziter Typ 183
Implizites COMMIT
 Auswirkung 642
Implizite Typdefinition 183
Import
 zu Export 259
Importieren
 Klasse 359
Import markieren 359
Importparameter
 festlegen 261
IN 611, 664
Inaktiv 138
IN BINARY MODE 681
Include 95
 Spezialisierung 256
INCLUDE 255
 Endung 256
 Ereignis 256
 TOP 256
 Unterprogramm 256
 UXX 256
INCLUDES 256
Index 562, 586, 664
 Pflegemaske 587
INDEX 186
Index anlegen 587
Indexfeld 588
Indextabelle 185, 195
Indexvariante
 löschen 196
Indexzugriff 185
Indirekte Adressierung 288
Informatiker-Geheimsprache 269
Information
 anfüttern 668
 Semantische 440
 Technische 440, 446
Information Hiding 238, 286
Infosystem 130, 131
Infotext 513
Infrastruktur 30
INITIAL 213
Initiale Platzgröße 576
INITIAL-Extent 576
Initialisierung und Zugriff 190
INITIALIZATION 233, 465
Initialwert 204, 245
 Spalte 571
Inner Join 616, 626
INNER JOIN 633
INPUT 237
INSERT 191, 192, 597, 636, 650
 Beispiel 651
 FROM TABLE 652
 Laufzeitfehler 652
 sy-subrc 650
INSERT INTO 650
INSERT REPORT 701
Instanz 161
Instanz-Constructor 307

Inst.-Erzeugung 373
INT4 171
 Ausgabe 199
Integer 163
Interaktionsübersichtsdiagramm 276
Interaktion von Mensch und
 Maschine 498
Interaktivität 435
Interface 375
 Anlagedialog 393
 Anlagetyp 393
 Anlegen 393
 Attribut 382
 Auflistung 385
 Definitionsreihenfolge 385
 Down-Cast 392
 Dynamischer Typ 391
 Element 382
 globale Klasse 395
 Globales 393
 IMPLEMENTATION 386
 Implementieren 384
 Karteireiter Attribute 394
 Karteireiter Interfaces 394
 Karteireiter Methoden 394
 Klasse Definitionsteil 383
 Klasse Implementierungsteil 383
 lokales 381
 Methode 382
 Methodenaufruf 388
 namensgleiche Methode 385
 Objektreferenz 391
 Service 379
 Serviceschnittstelle 384
 Sichtbarkeit 383
 Sichtbarkeitsabschnitt 384
 Statischer Typ 391
 Tilde 386
 UML 382
 Verwaltung 390
 Zusammengesetztes 394
INTERFACE 381
Interfacereferenz 391, 691
Interner Modus 136
 Speicherbereich 136

Interne Tabelle 182, 293
 ändern 196
 anhängen 191
 Arbeitsstruktur 191
 Art 185
 ausgeben 194
 befüllen 193
 Datenermittlung 186
 deklarieren 183
 DELETE 196
 Feldsymbol 689
 Kurzform 183
 löschen 196
 MODIFY 196
 Pufferung 610
 Schlüsseldefinition 183
 Spalte 183
 Tabellenart 183
 Zeilentyp 183, 184, 189
 Zusammenstellung 188
Internetanwendungen 40
Internetprotokolle 41
Internettechnologien 33
Internet Transaction Server 40
Interpreter 45
Interpretieren 45
Intervall 449
 Ausschließen 454
 ermitteln 209
 Selektieren 454
Intervallsuche
 Plan 209
IN TEXT MODE 681
INTO 206, 424
 ASSIGNING 690
 Einzelfeld 604
 interne Tabelle 604
 strukturiertes Feld 604
INTO CORRESPONDING FIELDS 604
INTO TABLE 611
INTO target 604
IS ASSIGNED 690
Isolation 637
ITS 40
Ivar Jacobson 276

J

Java 41
Java Script Object Notation 425
Jim Rumbaugh 276
JJJMMTT 209
Join 624, 629
 einleiten 633
 inner 616
 outer 616
Join-Bedingung 615, 617
Join-Verbindungstabelle 624
JSON → Java Script Object Notation

K

Kaffa 283
Kaffeebaum 306
Kaffeebohne
 Klasse 320
Kaffeekirsche 303, 308, 336
Kaffeevollautomat 268
Kahve 270
Kalender 440
Kapsel
 Unterprogramm 238
Kapselung 44, 228
Kardinalität 280
Katze
 Sofa zerkratzen 458
Keine Daten 101
Kellerspeicher 446
Kernabschnitt 144
Kettensatz 47, 97, 200, 293, 479
Key 92, 571
Kilokalorien 489
Kindklasse 342
Klammerausdruck 211
Klasse 288
 abstrakte 372
 aktivieren 320, 324
 Alias 387
 allgemeine 364
 Anlagedialog 321
 Assoziation 280
 Attribut 279, 295

- Klasse (Forts.)
 - Attributpflege* 322
 - definieren* 284
 - Definitionsteil* 284
 - dynamischer Typ* 364
 - finale* 321, 375
 - gewöhnliche* 321
 - GUI* 118
 - Implementationsteil* 284
 - implementieren* 284
 - Importieren* 358
 - Interface anbieten* 381
 - Interface-Auflistung* 385
 - Interface implementieren* 385
 - Interface-Methodenaufruf* 388
 - Karteireiter Interfaces* 395
 - Kästchen* 278
 - Kundennamensraum* 320
 - lokal* 284
 - Methode* 279
 - Methodennamensraum* 389
 - Namenskonvention* 321
 - object* 349
 - Objektyp* 288
 - Pfeilchen* 313
 - privat instanziierbare* 397
 - prüfen* 320
 - Service* 379
 - spezielle* 364
 - Statischer Typ* 364
 - Teilobjekt* 324
 - Testen* 320
 - Tilde* 386
 - Transparente* 390
 - Vererbungseigenschaft* 346
 - Zusammenarbeit* 280
 - Klassen-Constructor 308
 - Klassendiagramm 276, 278, 280
 - Bauplan* 281
 - Klassenhierarchie 347
 - Klassenkomponentenselektor 335
 - Klassenpyramide 346
 - Klassentyp
 - Ausnahmeklasse* 420
 - gewöhnliche ABAP-Klasse* 395
 - Klasse testen
 - F8* 320
 - Klausel 600
 - Klebezetzel 491
 - Kleinbuchstabenrelevant 87, 573
 - Kleiner 611
 - Kleiner gleich 611
 - Kloster Chéhodet 275
 - Kochrezept
 - cl_gui_alv_grid* 350
 - Kollaborateur 280
 - Kollege
 - Danke* 252
 - Komfort 521
 - Komma 480
 - Kommentar 48, 97
 - automatischer* 258
 - kommentieren 48
 - Kommunikation
 - Empfänger* 330
 - ereignisbasierte* 330
 - Kochrezept* 330
 - Sender* 330
 - Kommunikationsschritt 316
 - Kommunikationsstatistik 62
 - Kommunikationsstruktur 109
 - Kompatibel 203
 - nicht* 203
 - Kompilieren 45
 - Komponente 176, 178
 - auslesen* 698
 - Name* 179
 - Spalte* 184
 - Typ* 179
 - Typbezug* 178
 - Zugriff* 177
 - Komponententyp 179
 - Konkretisierung 161
 - Konstante 298
 - Kontextabhängige Suche 459
 - Kontextmenü 86
 - Entwicklungsobjekt* 143
 - Kontrollstruktur 214
 - Konvertibel 203
 - Konvertierung 205
 - Download* 676
 - Konvertierungsfehler 204
 - Konvertierungsregel 202, 203
 - Konzern 56
 - Kooperatives Programmieren 252
 - Kopfgesteuerte Schleife 221
 - Kopflastige Schleife 221
 - Kopfzeile 188
 - Arbeitsstruktur* 188
 - Kopiervorlage 161
 - Kreuzprodukt 616, 624, 629
 - Kühlen 371
 - Kunde
 - Attribut* 567
 - Kundennamensraum 320
 - Kundenstamm 567
 - Kundenverwaltung 567
 - Kunstgriff 243
 - Kürzel
 - dreistelliges* 462
 - KVA 268
- ## L
- Label
 - Text* 524
 - Lagerhalle 30
 - Länge 209
 - ermitteln* 205
 - fixe* 165
 - Lastverteilung 53
 - Laufzeit 186, 316
 - Variable* 159
 - Laufzeitauswertung 695
 - Laufzeitfehler 403
 - Abfangbarer* 414
 - dynamische Programmierung* 699
 - Rollback* 639
 - Systemreaktion* 416
 - verhindern* 653
 - Laufzeitsystem 97
 - Eingabeunterstützung* 440
 - Ereignisblock* 231
 - Laufzeitumgebung 45, 109, 134, 464
 - Läutern 357

- Layout 236, 488
- Layoutbereich 523
- Layoutdefinition 521, 529
- lcl_singleton 398
- LE 611
- Lebensdauer 283
- Lebenslinie 316
- Leerzeichen 47, 97
- Leerzeilen 225
- LEFT OUTER JOIN 633
- Lesen
 - Index* 195
- Lied 662
- LIKE 292, 439, 607, 611
- LIKE LINE OF 191, 293
- lines 631
- Linux-Downloads 33
- Listbreite 224
- Liste 97, 431, 477, 490
 - Farbe* 480
 - Gestaltung* 480
 - Icon* 478
 - Inhalt erzeugen* 465
 - Kopfzeile* 477
 - Leerzeile* 480
 - Listenüberschrift* 477
 - Seitennummer* 477
 - SKIP* 480
 - Symbol* 478
 - Titel* 477
 - ULINE* 480
- Listen 42
- Listenpuffer 97, 477
- Listenzeile 109
- Listüberschrift 462
 - pflügen* 478
- Literal 212
- Literalbegrenzer 97
- Live-Performance
 - Unterprogramm* 247
- Load 45, 122, 464
- LOAD 42
- LOAD-OF-PROGRAM 230, 233, 464
- Lochkarten 38
- Lochkartenlesemaschinen 39
- Logical Unit of Work 638, 642
- Logik im Fuß 217
- Logiktafel 212
- Logische Bedingung 102, 194, 214, 220
 - WHILE* 221
- Logische Datenbank 434
- Logische Gruppe 475
- Logische Prüfung 150
- Logischer Ausdruck 212, 611
 - verknüpfen* 212
- Logischer Dateiname 682
- Logischer Operator 212
- Logischer Speicherparameter 575
- Logisches Oder 611
- Lokale Datendefinition 310
- lokale Klasse 284
 - importieren* 358
 - Vererbung* 344
- Lokaler elementarer Typ 168
- Lokaler strukturierter Typ 176
- Lokaler Tabellentyp 182
- Lokaler Typ 354
- Lokales Interface 381
- Lokales Objekt 141
 - \$TMP* 142
- Lokale Typdefinition
 - technischer Aspekt* 171
- Lokale Überdeckung 246, 311
- Lokale Variable 245
 - namensgleiche* 246
- LOOP 192, 408
 - Feldsymbol* 690
- Löschen
 - mehrere Zeilen* 659
 - selektives* 660
- LOW 450
- LT 213, 611
- Lucky Case 402
 - Schrödinger* 430
- Lucky-Case-Passage 424
- M**
- mainframebasierte Business Suite 39
- Maischen 343
- Makro-Assembler 42
- Maltoserast 345
- Mandant 56, 570
 - automatisch versorgter* 633
 - Konzern* 56
- mandantenabhängig 570
- Mandantenfeld 570
- Mandantenkonzept 123, 570
- mandantenunabhängig 123, 570
- MANDT 125, 570
- Marie 641
- Marmorkuchen 641
- Maschinencode 464
- Massendaten
 - einfügen* 652
- Massenpflege 527
- Master-of-Objektiversum 285
- Master-Passwort 34
- Materialwirtschaft 40
- Matrix 491
- Matroschka-Prinzip 460
- m aus n 442
- Mauszeiger
 - Aussehen* 482
- Maximale Beglückung 439
- me 312
- Megaoperation 204
- Mehrfachselektion 129, 453, 474
- Mehrfachvererbung 342
- Mehrsprachige Anwendung 96
- mehrwährungs- und mehrsprachenfähig 39
- Membervariable 292
- MEMORY ID 448
- Menü
 - Hilfe* 61
 - System* 61
- Menüeintrag 61
 - expandieren* 63
 - komprimieren* 63
- Menüleiste 61, 508
 - Eingabefeld* 511
 - Funktion* 512
 - Hilfe* 512
 - Kontextmenü* 64
 - System* 512

- Menu Painter 490, 510, 511
 - Funktionstaste 517
 - Vorlage abgleichen 512
 - Menüpflege 511
 - Message 470
 - Nachrichtenklasse 471
 - Nummer 471
 - Platzhalter 471
 - Typ 471
 - WITH 471
 - Metabeschreibung 598
 - METHOD 300
 - Methode 219, 279, 299, 300
 - Aufruf 314
 - Aufrufvariante 315
 - Ausnahme 324
 - Constructor 307
 - Definition 301
 - dynamisch 700
 - Eigenschaft 373
 - EXCEPTIONS 305
 - EXPORTING 240
 - funktionale 305
 - Implementierung 301, 309
 - IMPORTING 240
 - Karteireiter 322
 - Kommunikation 327
 - Name 300
 - Namensraum 386
 - Pflege 323
 - Quelltext 324
 - Schnittstelle 304
 - Schnittstellendefinition 304
 - Semantik 361
 - Zugriff 310
 - Methodenaufruf 314
 - Methodenpflege 323
 - METHODS 300, 382
 - Microsoft-Loopback-Adapter 34
 - MIME Repository 139
 - Mission Impossible 418
 - Mitarbeiter
 - zuordnen 73
 - MOD 211
 - Modellierung 272, 281, 567
 - Bier 338
 - Vererbung 341
 - Model-View-Controller 487
 - MODIFY 196, 636, 661
 - ändern 661
 - Beispiel 196
 - dynamisch 697
 - einfügen 661
 - Indexvariante 196
 - INSERT 661
 - Schlüsselvariante 196
 - sy-dbcnt 662
 - Syntax 662
 - sy-subrc 661, 662
 - UPDATE 661
 - MODIFY CURRENT LINE 485
 - Modul 116, 236
 - anlegen 117, 531
 - Aufruf 117
 - Definitionsort 117
 - erzeugen 117
 - INPUT 237
 - keine Schnittstelle 237
 - Modul-Pool 539
 - OUTPUT 237
 - Programm 237
 - Programmieren 531
 - Rahmenprogramm 117
 - Unterprogramm 534
 - Modularisierung 97, 234
 - Modularisierungseinheit 237
 - MODULE 237, 502, 504
 - Anweisung 117
 - Modulen 40
 - Modul-Pools 42
 - Modul vs. Unterprogramm 237
 - Modus 65
 - Anzahl 65
 - Liste 66
 - neuer 57
 - Nummer 66
 - öffnen 65
 - schließen 66
 - Moduswechsel 98
 - möge ABAP mit dir sein 703
 - Mohammed 275
 - Morsecode 61
 - Motivationscoach 252
 - MOVE 42
 - MOVE-CORRESPONDING 604
 - Multiplikation 211
 - MULTIPLY 211
 - Murphy's Law 402
 - Mussfeld 107, 445
 - OBLIGATORY 445
 - Muster 149, 213, 263
 - Berechtigung 556
 - Raise Exception 422
 - Mustersuche 206
 - Mustervergleich 611
 - Mutterklasse 342
 - MVC 487
- ## N
- Nachguss 350
 - Nachkommastelle 167, 293
 - Nachricht 431
 - Nachrichtenklasse 420, 471
 - Name 471
 - Pflege 471
 - Nachrichtenkopf 428, 430
 - Nachrichtenkörper 428, 430
 - Nachrichtenkurztext 472
 - Nachrichtenlangtext 472
 - Nachrichtentyp
 - A, X 639
 - Namensgleichheit 535
 - Trick 112
 - Vergleich 611
 - Namenskonvention 162, 170, 321
 - Dictionary-Struktur 521
 - Vorteil 170
 - Namenspräfix
 - zif 393
 - Namensraum 77
 - Kunde 77
 - SAP 77
 - Native SQL 597, 598

Navigationsbereich 63, 72, 139
 ausblenden 143
 Breite 143
 Pfeil 144
 Sichtbarkeit 141
Navigationshistorie 144
Navigationspfeil 144
Navigationsverhalten 495
NE 213, 611
Neuen Modus erzeugen 136
Neue Zeile 479
 Schrägstrich 479
NEXT-Extent 576
Nibble 159
Nicht bedingte Schleife 217
Nicht case-sensitive 47
Nicht hierarchische Liste 349
Nicht-Schlüsselwort 148
Nicht-Unicode-System 682
Nico 582
NO-GAP 482
NON-UNICODE 682
NON-UNIQUE 187
Normalisierung 579
NOT 212, 611
NTFS-Dateisystem 33
Nudellängen- und Kalorien-
 Kalkulator 488
Nudelmenü 511, 513
Null-Pointer 289

O

Oberfläche 144
Oberklasse 342
object 349
Object Management Group 276
Object Navigator 64, 72, 122, 138,
 433, 493
 aufrufen 72
 Entwicklungsobjektliste 139
 Navigationsbereich 72, 139
 Objektliste 72
 Werkzeugbereich 72, 139
Objekt 282
 aktiv testen 138

anlegen 146
Attribut 295
CREATE PRIVATE 398
Ereignis 331
erzeugen 118
Hauptspeicher 283
inaktiv testen 138
Kommunikation 314
Komplexität 272
Lebensdauer 283
Navigation 144
PRIVATE 286
Privatsphäre 286
PROTECTED 286
PUBLIC 286
testen 146
Objektdatenbank 580
Objektinstanziierung 287, 289
Objektkatalogeintrag 88
Objektklasse 552
 Anwendung 552
Objektkomponentenselektor 312,
 335
Objektliste 72
 Drag & Drop 672
 Spaltenbreite 143
 Werkzeugbereich synchronisieren 145
Objektliste anzeigen 145
Objektlistenauswahl 142
Objektorientierung 44, 126, 268, 282
 Abstraktion 281
 Attribut 279
 Begriff 270
 Daten 271
 Datenkapselung 281
 Daten und Funktionen 274
 hat etwas 270
 Holistische Sichtweise 271
 macht etwas 270
 Methode 279
 Modellierung 272
 Namensgebung 282
 Polymorphie 281
 Vererbung 281
 Verhalten 271
 Verkuppeln 333

Objektreferenz 219, 288, 691
 Globale Klasse 363
 TYPE REF TO 288
Objektrelationale Datenbank 580
Objektschlucker 329
Objektschrott 329
Objektschutz 77
Objekttyp 288
OBLIGATORY 104, 105, 445, 645
 Fehlermeldung 105
OCCURS 188
o-Deklination 509
Offset 209
 Startposition 209
ok_code 530, 531
OK-Feld 527
Oktett 159
O(n) 186
ON 633
One-Way Parser 345, 385
Online-Hilfe
 Konvertierungsregel 203
OO-Klasse 252
OO-Notation 276
OPEN DATASET 681, 682, 685
 FOR APPENDING 681
 FOR INPUT 681
 FOR OUTPUT 681
 Syntax 681
openlibrary.org 425
Open SQL 100, 595, 597
 Anweisung 597
 INSERT 650
 Kommando 645
 Operation 636
Operation 161
 Auswertungsreihenfolge 211
 Relation 563
Operator
 Division 211
 logischer 212
 Vergleich 213
 Zeichenketten-Vergleich 213
optimierter Zugriff 188
OPTION 450
OPTIONAL 304

OR 212, 664
 SELECT 611
 View 627
ORDER BY 608
Osterei 532
OTHERS 407
Outer Join 616
OUTPUT 237, 682

P

PAI 116, 236, 497
 Modul 237
 Passender Zeitpunkt 531
Paket 122
 \$TMP 142
 anlegen 77, 79
 Anlegeprozess 80
 Anwendungskomponente 78
 Art 77
 Container 77
 Detail 80
 Entwicklungsobjekt 127
 Kapselung 127
 Kontextmenü 86, 89, 91, 95
 Kurzbeschreibung 78
 lokal 142
 Paketname 78
 permanente Zuordnung 77
 schachteln 77
 Schnittstelle 77
 Softwarekomponente 78
 Verwendungserklärung 77
 zeitlich unbegrenztes 127
Paketname 78
Pakettyp 78
Parameter 104
 Call-By-Reference 304
 Call-By-Value 304
 CHANGING 305
 Eingabefeld 436
 Eingabeprüfung 436
 EXPORTING 305
 ID 447
 IMPORTING 305

Karteireiter Methoden 322
Name 104
OPTIONAL 304
 optionaler 259, 261
RETURNING 305
SELECT 106
 Selektionstext 104
 Typ 104, 439
 typisieren 244, 259
 USER-COMMAND 469
 VALUE 304
 Variable 436
 verpflichtender 436
 Vorschlagswert 259, 261
PARAMETER 240
Parameter anlegen
 Funktionsbaustein 259
Parameter aus der Parameterliste 310
Parameterliste 323
Parametername
 aussagekräftiger 438
 Präfix 438
 zu lang 438
PARAMETERS 97, 104, 133, 432,
 436, 490
 DEFAULT 439
 Deklaration 437
 Deklarationsvariante 436
 Drucktaste 645
 OBLIGATORY 645
 VALUE CHECK 646
PBO 116, 236, 497
PBO-Modul 237
PERFORM 243, 247, 299
 IN PROGRAM 701
Performancestatistik 53
Performantes Lesen 194
Persistente Daten 182
persistentes Objekt 397
Persistenz 83
Personal 40
Perückenform 166
Perückengröße 164
Pfad 669
 ermitteln 671

Pfadangabe 676
Pfeilchen 312, 313
 -> 312
Pflagedialog
 Transaktion 120
Pflege-View 615
 outer Join 616
Physischer Name 682
Physischer Speicherbereich 575
Physische Sperre 663
Plan 161
Planung
 Stichtagsbezogene 437
 Stichzeitbezogene 437
plattformabhängigen Interpreter 42
Platzhalter
 Suche 129
Platzverschwendung 206
POH 236
Pointer 288
Point of no Return 218
Polymorphia 355
Polymorphie 44, 355
 alternative Implementierung 355
Pooltabelle 594
Poorman's Syntax Check 47, 96, 149
POPUP_TO_CONFIRM 659
Positioniert
 UI-Element 647
Potenzieren 211
Potenzrechnung 211
POV 236
Präfix
 CX_SY_ 419
 so_ 451
 zcx_ 419
Präprozessor 42
Präsentationsschicht 40, 133
Präsentationsserver 236
Pretty Printer 47, 149, 324
 Einrückung 149
 Einstellung 149
 Groß- und Kleinkonvertierung 149
Primärindex 586
 sortiert abgelegt 586

- Primärschlüssel 92, 563, 571, 586, 650
 - definieren 92
 - Eindeutigkeit 563
 - zusätzliches Feld 572
 - Primitiv 168
 - PRIVATE 286
 - Privates Attribut 298, 353
 - Privates Instanzattribut 353
 - Process After Input 116, 236, 470, 497
 - Passender Zeitpunkt 531
 - Process Before Output 116, 236, 470, 497
 - Process On Help Request 236
 - Process On Value Request 236
 - Produktion 30, 40
 - Produktivsystem 73
 - Produktreihe 200
 - Professional User Transaction 540
 - Profil 550
 - Profilparameter 136
 - Programm 54
 - abschmieren 413
 - absichern 548
 - aktivieren 118
 - Analyse 150
 - analysieren 151
 - anlegen 95, 122, 146
 - ausführbares 54
 - ausführen 64, 133
 - Eigenschaft 95
 - entwickeln 98
 - Ereignisblock 229
 - interner Modus 136
 - Klasse 317
 - Laufzeit 134
 - Liste 98
 - Load 464
 - mandantenunabhängiges 124
 - Modul 237
 - modularisieren 234
 - Oberfläche 133
 - Objekt 136
 - Quelltext 462
 - Report 42
 - Repository 702
 - Ressource 134
 - schützen 548
 - Skelett 199
 - Stundenberechnung 151
 - suchen 128
 - TOP-Include 95
 - Programmabschnitt
 - Klammern 414
 - Programmausführung
 - Debugging 150
 - schrittweise 154
 - Programmbibliothek 128
 - Programm 128
 - Programme erzeugen 700
 - Programmentwicklung
 - Datenhaltung 562
 - Programmerzeugung
 - im Hauptspeicher 700
 - Programmerzeugung im Repository 701
 - Programmgeschichten 47
 - Programmieren
 - SE80 64
 - Programmiererkarriere 549
 - Programmierschnittstelle
 - Datenbank 596
 - Programmiersprache 41
 - Programmierwerkzeugkiste 695
 - Programming by Difference 343, 365
 - Programmkopf 95
 - Programmlokal
 - Wiederverwendung 228
 - Programmlokaler Typ 162
 - elementarer 168
 - Programmstruktur 102
 - Programmtitel 477
 - Programmtyp
 - ausführbares Programm 539
 - Modul-Pool 539
 - Programm und Dynpro (Dialogtransaktion) 539
 - Programm und Selektionstyp (Reporttransaktion) 539
 - Programmzeile 96
 - Projektion 615, 621, 629
 - Projektions-View 615
 - Projektleiter 73
 - Auftrag freigeben 74
 - Projektorganisation 74
 - Prominenz 511
 - PROTECTED 286
 - Protokoll 181, 681
 - HTTP 425
 - Protokolltabelle 577
 - Prozedural 271
 - Prozedurales ABAP 267
 - Prozesse 41
 - Prüftabelle 579, 583, 584
 - Prüfung
 - deaktivieren 558
 - Prüfwert 580
 - PUBLIC 286
 - Puffer
 - synchronisierter 597
 - vorbeilesen 632
 - Pufferabgleich 597
 - Puffermanagement 597
 - Pufferung 576, 577
 - Pufferungsart 576, 577
- ## Q
- Qahwa 270
 - Qualitätssicherung 30
 - Quelltext
 - hoppeln 155
 - Quelltexteditor 324
- ## R
- R/1 38, 42
 - R/2 39, 42
 - R/3 40
 - Radiobutton 443
 - Auswahl 469
 - RADIOBUTTON GROUP 443
 - Radiobutton-Gruppe 443
 - Name 443
 - Rahmen 460, 490, 524

- Rahmenprogramm 234, 256, 492
 - anzeigen* 255
 - RAISE EVENT 331
 - RAISE EXCEPTION 422
 - RAISING 421
 - Randwert 215
 - Rational 276
 - RDBMS 596
 - Read 636
 - READ
 - Index* 195
 - WITH* 195
 - Zielstruktur* 195
 - READ DATASET 681, 685
 - Read-Only 322
 - READ-ONLY 351
 - READ TABLE 195, 408
 - dynamisch* 698
 - Feldsymbol* 690
 - READ TEXTPOOL 466
 - Real-time data processing 38
 - Rechenkünstler 211
 - Redefinieren 355, 361
 - Redefinition
 - aufheben* 362
 - Drucktaste* 362
 - Globale* 358
 - lokale* 358
 - REDEFINITION 356
 - Redefinitionsdrucktaste 356
 - Redundanz 228
 - vermeidbare* 579
 - vermeiden* 614
 - Referenztyp 293
 - Referenzübergabe 240
 - Registrierung 333
 - Reihenfolge
 - Aktualparameter* 247
 - Reinheitsgebot 340
 - Relation 563
 - Relationale Algebra 563
 - Relationale Datenbank 83
 - Relationales Datenbanksystem 563
 - REPLACE 208
 - Report 42, 54
 - ausführen* 54
 - REPORT 105
 - Reporttransaktion 539
 - Repository 122, 123, 464
 - Kunde* 125
 - SAP* 125
 - Repository Browser 122, 139, 141, 255
 - Funktionsgruppe* 255, 257
 - Funktionsgruppe anlegen* 257
 - Objektliste* 142
 - Objektname* 142
 - Suchfehler* 141
 - Repository-Daten 57
 - Repository-Infosystem 128, 130, 139, 551, 568
 - Programmbibliothek* 128
 - technischer Aspekt* 128
 - Repository-Objekt 123
 - anzeigen* 139
 - mandantenunabhängiges* 124
 - REPOSRC 124
 - Request 427, 428
 - Reservierter Bereich 115
 - Response 427, 428
 - Objekt* 430
 - Ressourcennutzung 134
 - Rest 211
 - Returncode 222
 - RETURNING 305
 - Rezeptprogramm 224
 - Ringenspiel 498
 - Rohkaffee 334
 - Rollback 638
 - expliziter* 639
 - impliziter* 639
 - ROLLBACK WORK 639
 - Rolle 550
 - Rote-Beete Saft 328
 - RTTC 692
 - RTTI 467, 692
 - RTTS 692
 - Rückgabecode
 - Clipboard* 674
 - run 218
 - Rundung 167
 - Runtime 464
 - Runtime Type Creation 692
 - Runtime Type Identification = RTTI 692
 - Runtime Type Services 692
- ## S
- SAP
 - anmelden* 52
 - Datenbank* 563
 - Desktop* 52
 - Einführung* 29
 - Gründung* 37
 - im Einsatz* 28
 - Oberfläche* 431
 - Programmliste* 52
 - Unternehmen* 37
 - Willkommenstext* 58
 - SAP_ALL 50
 - SAP All-in-One 41
 - SAP-Anmeldung 55
 - SAP-Anwender
 - verteilen* 53
 - SAP Basis 44
 - SAP Business One 41
 - SAP Business Suite 41
 - SAP Community Network 31, 34
 - registrieren* 31
 - SAP-Directory 683
 - SAP Easy Access 63
 - Einstellung* 64
 - SAP ERP 41
 - SAP GUI 29, 30, 52, 56, 133, 431
 - Aufbau* 61
 - aufrufen* 36
 - Client* 55
 - Funktion* 61
 - Für Windows* 540
 - Instanz* 65
 - Programmierung* 490
 - SAP GUI-Fenster 65
 - SAP HANA 580
 - SAP-Internetstrategie 40
 - SAP Logon 36, 52
 - Desktop-Symbol* 52
 - Knoten Verbindungen* 52

- Programmliste 52
- SAP-System 52
- starten 52
- Verknüpfung 54
- SAPLSU_USER 256
- SAP-LUW 637, 642
 - SAP-Sperre 663
- SAP Management Console 36, 43
 - starten 36
- SAP-Memory 436, 446, 448
- SAP-Menü 63
- SAP NetWeaver 44
- SAP NetWeaver Application Servers
 - ABAP 44
- SAP NetWeaver Main Releases 31
- SAP NetWeaver-Technologie 41
- SAP R/3 Enterprise 41
- SAP-Sperre 663
 - freigeben 663
 - setzen 663
- SAP-Sperrkonzept 640
- SAP-System 52
- SAP Systemanalyse und Programm-entwicklung 37
- SAP-Temp
 - Verzeichnis 671
- SAP-Transaktion 54, 135, 637, 643
 - SAP-LUW 643
 - SAP-Programm 643
- SAP-Transaktionen 29
- SAP-Transaktionskonzept 637
- SAP Web Application Server 41, 44
- Satzaufbau 198
- Schablone 161
- Schicht 132
- Schlauchen 376
- Schleife
 - EXIT 217
- Schleife durchlaufen
 - Bedingung 217
- Schleifen 30
- Schleifenanatomie 217
- Schleifendurchlauf 406
- Schleifenfuß 217
- Schleifengrundkonstruktion 217
- Schleifenindex 222
 - LOOP 222
- Schleifenkonstrukt 217
- Schleifenkopf 217
- Schleifenkörper 217
 - abarbeiten 220
- Schleifenzähler
 - sy-index 220
- Schlüssel 187, 190, 563
 - Eindeutigkeit 187
 - einzelne Komponente 187
 - ganze Zeile 187
 - Reihenfolge 187
- Schlüsselangabe
 - löschen 196
- Schlüsselart 190
- Schlüsseldefinition 183, 565
- Schlüsseleigenschaften
 - ABAP 46
- Schlüsselfeld 195
 - bündeln 620
- Schlüsselfelddaten 586
- Schlüsselkomponente 187, 190
- Schlüssellesen 195
- Schlüsselsymbol 581
- Schlüsselwörter 46
- Schlüsselwort groß 141, 149
- Schlüsselzugriff 185
- Schnittstelle 238
 - generisch typisierte 186
 - große Datenmenge 245
 - Problem 243
 - Typisierung Tabelle 186
- Schnittstellendefinition 239
- SchnittstellenPARAMETER 240
- Schrägstrich 479
- Schreibmaschine 147
- Schriftart 148
- Schritthöhe 164
- Schrödinger-Rezept 198
- Schrödinger-Tasse 303
- Schrödinger-Weiße 340
- Schublehre 99
 - ausführen 99
- Schüsselspalte 195
- Schutz 663
- Schwaiger Roland
 - bisschen verrückt 503
 - Klebezettel-Notation 435
- SCN 31, 34
- Screen
 - Variante 463
- Screen Painter 110, 490, 494, 522
 - Ablauflogik 116
 - Drag & Drop 522
 - Eigenschaft 111
 - Einstellung 522
 - Folge-Dynpro 111
 - Grafischer Layout-Editor 522
 - Kurzbeschreibung 111
 - Layoutbereich 522
 - Textfeld 524
 - Werkzeugleiste 522
- SE11 404
- SE24 320
 - Klasse anlegen 321
 - Redefinieren 361
- SE37 411
- SE80
 - verwenden 79
- SE91 420, 421
- SEARCH 206
- Sekundärindex 586
 - Bezeichnung 587
 - erstellen 587
 - Feld 588
 - Kurzbeschreibung 587
 - Nachteil 586
 - Ziel-DB 587
- Sekundärindexpflege 587
- Selbstreferenz 312, 317, 335, 351
 - me 312
- SELECT 100, 242, 597, 600
 - * 601
 - Abschnitt 600
 - alle Spalten 601
 - Beispiel 609
 - DISTINCT 601
 - Dokumentation 100
 - dynamisch 697, 698

SELECT (Forts.)
 dynamisches Token 699
 ENDSELECT 602
 Ergebnis 100
 Ergebnismenge 100
 FROM 605
 GROUP BY 608
 HAVING 608
 IN 611
 INTO target 604
 Klammer 611
 Klausel 601
 NOT 611
 OR 611
 ORDER BY 608
 Schleife 612
 SELECT-OPTIONS 610
 SINGLE 100, 601
 sy-dbcnt 610
 sy-subrc 404, 610
 View 615
 View-Daten 633
 WHERE 100
 SELECTION-SCREEN 460
 SELECTION-SCREEN BEGIN OF 435
 SELECTION-SCREEN
 PUSHBUTTON 647
 SELECTION-SCREEN SKIP 646
 Select-Option
 Bedingung 455
 CHECK 457
 Datenobjekt 450
 Excluding 455
 FOR 451
 HIGH 450
 IN 456
 Including 455
 Interne Tabelle mit Kopfzeile 450
 LOOP 457
 LOW 450
 Menge von UI-Elementen 450
 Obergrenze 450
 OPTION 450
 SELECT 456
 Selektionsmenge 455
 SIGN 450
 Suchbedingung 129
 Untergrenze 450
 Vergleichsoperator 455
 Verwendung 129
 SELECT-OPTIONS 107, 432, 449,
 451, 490
 Intervall 107
 SELECT result 601
 SELECT-Schleife
 schachteln 664
 SELECT SINGLE 103
 Selektion 615
 WHERE 607
 Selektionsbedingung
 View 627
 Selektionsbild 71, 104, 108, 431, 490
 definieren 104
 Drucktaste 645
 Gestaltung 434
 leeres 107
 Spartanisches 438
 Selektionsbildprogrammierung 694
 Selektionsergebnis
 Einzelsatz 601
 Tabelle 601
 Selektionskriterium 129, 449, 629
 hinterlegtes 129
 Selektionstext 105, 439, 440,
 611, 647
 definieren 105
 Dictionary-Referenz 611
 Selektionstextpflege 105
 Semantische Eigenschaft
 Fremdschlüssel 583
 Semantische Information 440
 Semantischer Aspekt 171
 Dokumentation 171
 Feldbezeichner 171
 Hilfe 171
 Semmerl 553
 sender 218, 332
 Sender 330, 331
 SEPARATED BY 206
 Sequenzdiagramm 276, 278, 316
 Kommunikationsschritt 316
 Lebenslinie 316
 Objekt 316
 Server 55
 Serverinfo-Feld 62
 Serverinformation 62
 Service 378
 Realisieren 379
 Umsetzung 379
 serviceorientierter Architektur 41
 Servicerealisierung 378
 Session 136
 SET HANDLER 333
 SET-Parameter 447
 SET SCREEN 533
 Setter 279
 SET TITLEBAR 464, 506
 Shared Memory 597
 Puffer 597
 SHIFT 208
 Shortcut 54
 Sicherheitsabfrage 659, 679
 Sicherheitsabteilung 30
 Sichtbar
 lokal 245
 Sichtbarkeit 318
 Sichtbarkeitsbeschränkung 286
 SIGN 450
 Signatur 247, 324, 356, 365
 Unterprogramm 247
 Singleton
 get_instance 397
 Nutzmethode 399
 Referenz ermittelt 399
 Typ 399
 UML 397
 Singleton-Party 397
 Singleton Pattern 397
 Sitzung 136
 Skalierung 135
 SKIP 225, 435, 465, 480
 Skizze 519
 SOA 41
 Softwarehistoriker 404

- Softwarekomponente 78
 - HOME 78
- Softwarekrise 282
- Softwarequalität 338
- Solmisation 312
- SORT 192
 - ASCENDING 192
 - DESCENDING 192
 - dynamisch 698
 - Sortierreihenfolge 192
- Sorted 185
- SORTED 186
 - Feldsymbol 690
 - INSERT 192
- Sortieren 187
- Sortierte Tabelle 185
- Sortierung 194
 - absteigende 195
- Spaghetti 228, 488
- Spaghetticode 27, 431
- Spalte 563
 - Beschreibung 84
 - Datenelement 92
 - definieren 92
 - Key 92
 - Name 83
 - Schlüssel 83
 - Spalte 655
 - Transport 611
 - Typ 83, 84
- Spaltenformat 676
- Spaltenname 92
- Spaltenorientierte Speicherung 580
- Spaltenselektor 633
- Speedreading 144
- Speicherbereich
 - zugreifen 691
- Speichern
 - persistentes 83
- Speicherorganisation 135
- Speicherplatz 165, 578
- Speicherplatz
 - reservieren 690
- Speicherung 562
- Speisezugabe 376
- Sperrbaustein 599, 663
- Sperrre
 - entfernen 663
 - physische 663
- Sperrkonzept 599, 663
- Sperrobjekt 663
 - anlegen 663
- Spezialisierung 342
- SPLIT 207, 208
 - Tabelle 207
- Sprachenteam 42
- Sprachschatz
 - ABAP 161
- Springpapier 63
- Sprungmarke 97
- SQL 563, 566, 596
 - BYPASSING BUFFER 605
 - CLIENT SPECIFIED 605
 - FROM 605
 - Native SQL 597
 - Open SQL 597
 - Struktur auffüllen 604
 - UP TO ROWS 605
 - View 605
 - WHERE 607
 - Zielbereich 604
- SQL-Anweisung 565
- sscrfields 645
- ST22 370
- Stammdaten 575
- Standalone- Programmiersprache 50
- Standard 185
- STANDARD 183, 186
 - anhängen 192
 - APPEND 191
- Standard-ANSI-SQL 100
- Standardisierung
 - SQL 596
- Standardpaket 77, 127
 - kein Hauptpaket 80
- Standardschlüssel 187
- Standardselektionsbild 104, 434, 490
 - 1000 104, 437
 - Beispiel 611
 - Deklaration 104
- Standard SQL
 - Untermenge 597
- Standardtabelle 185, 190
- Startadresse 159
- START-OF-SELECTION 97, 229, 230, 233, 465, 477, 493
 - Startpunkt 389
- Startposition 209
- Startwert 245
- StarUML 279
- STATICS 218
- Statische Methode 335
- Statischer Typ 165, 364, 365
 - Bestellung 364
- Statisches Attribut 296, 313
- Statische Typisierung 166
- Status 57, 509
 - aktivieren 516
 - Aktivitätszustand 509
 - anlegen 510
 - Dialogfenster 510
 - Dialogstatus 510
 - Kontextmenü 510
 - Pflegedialog 509
- Statusfeld 57, 62
- Status-Icon
 - Erzeugen 410
- Statusinfo 57
- Statusinformation 62
- Statusleiste 62
- Statustyp 510
- S_TCODE 548
- Stefan Ehret 693
- Stereotyp 382
- Stichtagsbezogene Planung 437
- Storyboard 519
- Strg + V 674
 - testen 675
- Strichmännchen 482
- Strickmuster für Entwickler 397
- string 165
- stringtab 207
- STRLEN 205
- Structured Query Language 563, 566, 596

- Struktur 100, 163, 191
 - aktivieren 180
 - anlegen 178
 - Erweiterungskategorie 180
 - fehlende 228
 - Feld 102
 - flache 181
 - geschachtelte 181
 - Komponente 176, 178, 194
 - Kurzbeschreibung 178
 - SYST 530
 - tiefe 181
 - Typisierung 102
 - Zugriff 177
 - Zugriffsoperator 102, 177
 - Strukturausgabe
 - Problem 199
 - Strukturdatenobjekt 293
 - Strukturdiagramm 276
 - Strukturierter Typ 163
 - Strukturiertes Feld 604
 - Strukturpaket 77, 127
 - Strukturtyp 176, 184
 - lokaler 177
 - lokaler definierter 176
 - Name 176
 - Struktur 184
 - transparente Tabelle 184
 - View 184, 630
 - Strukturzugriff 200
 - Stuhl
 - Bauplan 161
 - Subklasse 342
 - SUBMIT
 - dynamisch 700
 - Subscreen 496
 - SUBTRACT 211
 - Subtraktion 211
 - Suchaufwand 664
 - Suche
 - Anwendungshierarchie 130
 - Berechtigungsobjekt 552
 - Kontextabhängige 459
 - Nachname 586
 - Platzhalter 129
 - Suchhilfe 452
 - Suchkomplexität 186
 - Suchkriterium 129
 - ausschließendes 129
 - einschließendes 129
 - Suchmöglichkeit
 - feingranulare 128
 - Suchvariante 688
 - super 356, 361
 - super->constructor 357
 - Superklasse 342
 - Super-Taste 128
 - Suppenbrunzer 486
 - sy-cprog 466
 - sy-datum 222
 - sy-dbcnt 100
 - SELECT 610
 - sy-index 217, 222
 - sy-langu 222, 466
 - sy-lisel 485
 - sy-lsind 484
 - sy-mandt 222
 - Symbol
 - Checkbox 482
 - symbolischer Name 689
 - Symbolleiste 61, 508, 517
 - abbrechen 61
 - Befehlsfeld 61
 - Enter 61
 - Layout anpassen 61
 - sichern 61
 - verlassen 61
 - zurück 61
 - Synchronisation 145
 - links nach rechts 145
 - rechts nach links 145
 - Synchronisierung 529
 - Syntaktische Korrektheit
 - prüfen 146
 - Syntax
 - grundlegende 96
 - Syntaxcheck
 - Schnittstelle 244
 - Syntaxchecker 150, 198
 - Interface 386
 - prüfen 201
 - syst 404, 530
 - System 512
 - Entwicklung 73
 - Produktion 73
 - Qualitätssicherung 73
 - Status 501
 - Systemadministrator 56
 - Systembefehl
 - ausführen 54
 - Systemfeld 222, 404
 - Auswahl 222
 - Ergebniswert 404
 - syst 404
 - sy-tabix 406
 - Systemglobal
 - Wiederverwendung 228
 - Systemkern 43
 - Systemlandschaft 73
 - Systemnachricht 62
 - Fehler 62
 - Information 62
 - Warnung 62
 - System RF 38
 - Systemschicht 132
 - Systemvariable 100
 - View 627
 - sy-subrc 100, 215, 222
 - Achtung 223
 - Ausprägung 405
 - AUTHORITY-CHECK 550
 - SELECT 610
 - sy-tabix 222, 406
 - LOOP 612
 - verwenden 226
 - sy-ucomm 473, 530
 - sy-uname 222
 - sy-zeit 151, 222
 - Format 151
- ## T
- T000 581
 - T100 420, 471
 - Tabelle 562, 601
 - abarbeiten 192
 - Abbildung 563
 - aktivieren 83, 584, 590

- am Join beteiligte* 633
- Auslieferungsklasse* 83
- Datensatz* 563
- eindeutige* 571
- Eingabehilfe/-prüfung* 584
- Eintrag erfassen* 199
- Feld* 570
- gepufferte* 598
- Inhalt* 593
- Kochrezept* 589
- Mandant* 570
- mathematische Beschreibung* 563
- mit Hut* 451
- Platzbedarf* 575
- Primärschlüssel* 563
- prüfen* 589
- puffern* 83, 664
- Schlüssel* 83
- sortieren* 192
- Spalte* 563
- T100* 471
- technische Einstellung* 83, 575
- Zeile* 563
- Zugriffsverhalten* 575
- zweidimensionale* 563
- Tabellenart 183, 190
 - Hashed* 185
 - Index* 185
 - Sorted* 185
 - Standard* 185
- Tabellenartiger Typ 163
- Tabellen-Cluster 594
- Tabelleneintrag
 - erzeugen* 192
- Tabellenfeld 570
- Tabellen-/Feldname 112
- Tabelleninhalt 592
 - optimierter Zugriff* 188
- Tabelleninhalt anzeigen 584
- Tabellen/Join-Bedingung 619
- Tabellenkalkulationsprogramm 676
- Tabellenkorpus 188
- Tabellenoperation 191, 196
- Tabellenpool 594
- Tabellenpuffer 576
- Tabellenpufferung 566
- Tabellensymbol 584
- Tabellentyp 163, 182, 189
 - anlegen* 694
 - eingebauter Typ* 189
 - globaler* 182, 189
 - lokaler* 182
 - Referenz* 189
 - Schlüssel* 187
 - Spalte* 184
 - Strukturtyp* 189
 - verwenden* 191
- TABLE
 - Schlüssel* 195
- TABLES 109, 530
 - Arbeitsstruktur* 645
- Tabula-Razor 204
- Tabulator 676
- Tagesdatum 209, 222
- Tastenkombination 61
- TCCODE
 - Feld TCD* 549
- Technik
 - Modularisierung* 252
- Technische Einstellung
 - Pflege* 575
 - sichern* 93
- Technische Information 440, 446
- Technische Literale 696
- Technischer Name 64
- Technischer Text 462
- Teilbaum markieren 130
- Teilbereich
 - aufklappen* 142
 - zuklappen* 142
- Teilobjekt 146
- Teilstring
 - ausschneiden* 209
 - Position* 209
- Temp-Verzeichnis
 - Client* 671
- Test
 - ausführen* 253
- Testen 146
- Testmodus 252
 - Funktionsbaustein* 253
- Testprogramm 142
- Testrahmen 411
- Testsystem 58
- Text
 - technischer* 462
- Textelement 440, 462, 611
 - Listüberschrift* 462
 - Selektionstext* 462
 - Textsymbol* 462
- Textfeld
 - beschreiben* 208
- Textkonserve 95
- Textliteral 97
- Textmodus 685
- Textmuster 206
- Textpool 461
- TEXTPOOL 466
- Tiefe Struktur 181
- Tilde 386, 633
 - eindeutiger Namensraum* 386
- Tim Berners-Lee 40
- Timepicker 458
- Timer 218, 219
 - erzeugen* 219
- tims 440
- Tipps & Tricks 483
- Titel 460, 502, 505
 - Anlageprozess* 506
 - Name* 505
 - Setzen* 504
 - Technisches Literal* 506
 - Titelleiste* 508
 - Werkzeugbereich* 145
- Titelleiste 61
- TITLE 460
- Token 695
- TOP
 - INCLUDE* 256
- Top-down-Ansatz 238
- TOP-Include
 - Datendeklaration* 95
 - Typdefinition* 95
- Top-Liste 195
- TOP-OF-PAGE 233, 479
- TOP-OF-PAGE DURING LINE-SELECTION 485

- Transaktion 42, 54, 120, 539, 541
 - ABAPDOCU 131
 - AL11 683
 - anlegen 120
 - Anlegen 539
 - Ausführbares Programm 539
 - Berechtigungsprüfung 548
 - FILE 682
 - GUI-Klassifikation 120
 - ICON 411, 669
 - Mit Kontextmenü anlegen 539
 - Professional User Transaction 540
 - schützen 663
 - SCU3 577
 - SE09 73, 75
 - SE11 86, 156, 170, 566, 618
 - SE16 569
 - SE16 693
 - SE37 252
 - SE38 95
 - SE51 110, 494
 - SE80 64, 72, 122, 137, 493, 702
 - SE81 130, 131
 - SE84 128, 131, 551, 568
 - SE91 420
 - SE93 540
 - SF01 682
 - SF07 682
 - starten 57, 64
 - Startobjekt 120
 - STMS 78
 - SU01 56, 500
 - SU21 551
 - SU53 559
 - Transaktionscode 120, 539
 - aktuelle Anwendung 57
 - anlegen 120
 - aufrufen 120
 - /n 57
 - Name 120
 - /o 57
 - SAP-Transaktion 643
 - SE61 58
 - Shortcut 54
 - suchen 131
 - technischer Name 64
 - Transaktion SE80 433
 - Transaktion SE91 471
 - Transaktionskonzept 636
 - Transaktionspflege 540
 - Transaktionssystem 637
 - TRANSFER 683
 - Transferstatus
 - Download 676
 - Transiente Daten 182
 - TRANSLATE 208
 - Transparente Tabelle 83, 564, 565
 - aktivieren 564
 - Auslieferungsklasse 568
 - Beispiel 567
 - Daten pflegen 569
 - definieren 568
 - Kurzbeschreibung 568
 - Nutzfelddefinition 565
 - Schlüsseldefinition 565
 - SM30 569
 - technische Einstellung 565
 - Verknüpfung 565
 - Transport 73, 122
 - Transportfeld 527
 - Transportschicht 74, 78
 - Transport- und Management-
system 122
 - Transportwesen 30
 - Treber 357
 - Treffermenge 664
 - Trennzeichen 207
 - TRY 371, 424
 - Turnsaal 318
 - Typ 158
 - c 169
 - d 169
 - Datenreferenz 691
 - eingebauter 161
 - elementarer 163
 - f 169
 - fehlender 243
 - globaler 161, 163
 - i 169
 - interne Tabelle 182
 - n 169
 - Name 162
 - optimale Speicherung 167
 - p 169
 - programmlokaler 161, 162
 - Rundung 167
 - string 169
 - strukturierter 163
 - t 169
 - tabellenartiger 163
 - unvollständiger 162
 - Variable 97
 - vererben 243
 - x 169
 - xstring 169
 - Typangabe 160
 - TYPE 104, 160, 162, 292
 - TYPE ANY 692
 - TYPE HANDLE 694
 - Typen 30
 - TYPE-POOLS abap 441, 442
 - TYPE-POOLS icon 478
 - TYPE-POOLS sym 478
 - TYPE REF TO 288
 - TYPE REF TO DATA 691
 - TYPES 162, 354
 - TYPE TABLE 183
 - Typgerechter Initialwert 204, 245
 - Typgruppe 442
 - abap 201, 630
 - ABAP 441
 - Icon 467, 669
 - Konstante 442
 - Namenskonvention 467
 - Typ 442
 - typisierten Datenspeicher 691
 - Typisierung 201, 243
 - kein Laufzeitfehler 244
 - Typkompatibel 171
 - Typname 160
- ## U
- Überdeckung 246
 - Überladen 356
 - Überprüfen
 - logische Bedingung 221
 - Überschneidung 166

Überschreiben 356
Überschrift 435
Übersetzbarkeit 462
Übersetzung 475
Übertragene Datenmenge 664
u-Deklination 509
ULINE 209, 435, 465, 480
UML 276, 278
- 298
+ 298
Aggregation 280
Analysewerkzeug 276
Assoziation 280
Bild 278
CRC 279
Dokumentation 278
dynamische Sicht 278
Kommunikationsgrundlage 278
Kunstwerk 278
Singleton 397
Standard 276
statische Sicht 278
Unterstreichung 296
UML-Sequenzdiagramm 315
Umsetzung
Service 379
Unabhängigkeitserklärung 564
Unautorisiert 548
Unautorisierte Verwendung 548
Unbedingter Haltepunkt 150
Ungleich 611
Unicode 44
Prüfung 180
Unicode-System 682
Unified Modeling Language 276
Uniform Resource Identifier 428
UNIQUE 187
UNIQUE DEFAULT KEY 187
Unterklasse 342
Sichtbarkeit 351, 352
Unterprogramm 193, 237, 238, 239,
299, 481, 534
Aufruf 238, 243, 247, 299, 314
Bündelungstechnik 643
CHANGING 240, 299

Datenübergabe 243
Definition 238, 239, 299
Doppelklick 238
dynamisch 700
ENDFORM 299
ersetzen 265
finden 248
FORM 299
Funktionalität 238
Funktionsbaustein 252
Implementierung 239, 299
Information Hiding 238
lokale Variable 238
Name 239, 247
Parameter 240, 299
Parameter definieren 242
PERFORM 299
Schnittstelle 238, 239, 299, 304
Schnittstellenparameter 240
Signatur 247
USING 240, 299
Verwendung 299
Werteübergabe 240
unterprogrammlokal 246
Unterzucker 353
Up-Casting 365
Update 636
UPDATE 597, 636, 655
dynamisch 697
SET 655
sy-dbcnt 655, 656
sy-subrc 655, 656
WHERE 655
Upload 666, 679
UP TO ROWS 605
URI 428
URL 118
USER-COMMAND 469, 647
USING 240, 299
Aufruf 247
praktisch 242
value 240
USR01 124, 367
UTF-8 682

V

VALUE 292, 304
VALUE CHECK 445, 646
Variable 100, 158, 159
ansprechen 165
Datenelement 172
Deklaration 97
deklarieren 98
direkte Adressierung 165
dynamische 165
erreichbare 229
ganzzahlige 202
globale 246
indirekte Adressierung 165
kein Zugriff 245
lokale 246
numerische 202
Startadresse 159
STATICS 218
statische 165
Struktur 100
strukturierte 177
Wert behalten 218
Zuweisung 205
Variablendeklaration 225
Variante
Anlegen 434
Vaterklasse 342
Verallgemeinerung 342
Veränderungen
Laufzeit 688
Verantwortlichkeit 278
Verarbeitungsblöcke 30
Verbindung
schließen 430
Verbuchung 643
Vererben 341
Vererbung 44, 338, 340
Attribut 360
cl_wd_button 348
Eigenschaft 343
Farbe 360
Globale/Lokale Klasse 363
Methode 360
praktisches Beispiel 348

- Vererbung (Forts.)
 - Syntaxchecker* 347
 - UI-Element* 348
 - Vererbungshierarchie
 - UI-Element* 348
 - Vererbungsstufe 343
 - Vergleichsoperator
 - LIKE* 607
 - textueller* 215
 - View* 627
 - Vergleichswert 195
 - Verhaltensdiagramm 276
 - Verknüpfung
 - definieren* 624
 - manuell definieren* 624
 - produzieren lassen* 624
 - Verkürzte Notation 188
 - Version
 - aktive* 138
 - inaktive* 138
 - Vertrieb 40
 - Vervollständigung 148
 - Verwaltungsobjekt 118
 - Verwendungsnachweis 155, 156
 - einschränken* 156
 - Rückwärtssuche* 155
 - starten* 156
 - Verzeichnisfunktion 667
 - Verzeichnissymbol 142
 - Verzweigungen 30
 - Verzweigungsliste 484
 - Überschrift* 484
 - Vielgestaltigkeit 355
 - Vier Tasten 145
 - View 605, 614
 - ABAP* 632
 - anlegen* 618
 - Beziehung* 624
 - Daten* 615
 - Feld* 615
 - Feld übernehmen* 620
 - Fremdschlüssel* 615
 - Inhalt* 623
 - Kurzbeschreibung* 619
 - programmieren* 630
 - Projektion* 615
 - Selektion* 615
 - Selektionsbedingung* 615
 - Tabelle* 619
 - Tabellenfeld* 619
 - Tabellen/Join-Bedingung* 624
 - technische Einstellung* 615
 - Verknüpfung* 623
 - View-Feld* 619
 - View-Daten 628
 - View-Feld 628
 - eindeutiges* 621
 - Name vergeben* 621
 - View-Typ 615
 - Virtueller Verarbeitungsblock 229
 - Visuelles Control 114
 - Vollbild ein/aus 143
 - Vorderwürze 357
 - Vorkommastelle 167
 - Vorlage
 - abgleichen* 512
 - Vorrangregel 211
 - Vorschlagswert 439
 - Vorwärtsnavigation 144, 146, 493
 - Domäne* 174
 - Modul anlegen* 531
 - Vorzeichen 167
- ## W
- Waagesymbol 99
 - prüfen* 99
 - Währung 167
 - WAIT 217
 - War of Notations 276
 - Wassermusik 60
 - Web 30, 545
 - Webanwendungen 33
 - Web Browser 33, 133
 - Web Dynpro ABAP 33
 - Werkzeug 122, 137
 - automatisches* 144
 - Werkzeugbereich 72, 95, 139
 - Objektliste synchronisieren* 145
 - Pfeil* 144
 - Titel* 145
 - Wert
 - gültiger* 201
 - zuweisen* 201
 - Wertehilfe 104, 236, 438, 458, 476, 584
 - Wertekopie 240
 - Werteprüfung 583
 - Werteübergabe 240
 - WHEN 103, 216
 - WHEN OTHERS 103, 216
 - WHERE 607, 664
 - AND* 611
 - dynamisch* 698, 699
 - WHILE 221
 - Durchlauf* 221
 - Wiederholung
 - Programm* 228
 - Wiederverwender 227
 - Wiederverwendung 71, 228, 516
 - programmlokale* 228
 - systemglobale* 228
 - Willkommensnachricht 239
 - Willkommenstext 59
 - WinRAR 34
 - WITH 195
 - WITH FRAME 460
 - WITH SMART LINEFEED 682
 - Workarea 191
 - Workbench-Auftrag 74, 75
 - Workflow 126
 - Work Process 134, 565
 - Datenbanksystem* 598
 - Puffer* 597
 - World Wide Web 428
 - Wort-Bedeutungsdilemma 277
 - WRITE 42, 96, 133, 200, 435, 465
 - COLOR* 480
 - dynamisch* 697
 - formatiert* 239
 - HOTSPOT* 482
 - Neue Zeile* 479
 - NO-GAP* 482
 - Schrägstrich* 479
 - Strukturausgabe* 199
 - write_field_separator 676
 - Würze 357

X

XSTRLEN 205
XUBNAME 367

Z

zcx_ 419
Zeichenfläche 523
Zeichenfolge
 zerteilen 207
Zeichenkette
 ausschneiden 209
 durchsuchen 206
 ersetzen 208
 komprimieren 208
 Länge 209
 Operation 209
 STRLEN 205
 suchen 208
 übersetzen 208
 verbinden 208
 Vergleich 213
 verknüpfen 205
 verschieben 208

 zerlegen 208
 Zugriff 209
Zeichenketten-Funktion 205
Zeiger 288
 Tabellenzeile 690
Zeile 563
Zeile für Zeile 192
 ausgeben 224
Zeilenende-Markierung 682
Zeilennummer 148
Zeilentyp 183, 184, 189, 293
 Möglichkeit 184
Zeilenumbruch 101, 200
Zeitpunkt 97, 229
Ziel-DB 587
Zielfeld
 zu kurzes 204
Zielstruktur 195
 Befüllung 604
Zuckerrast 350, 354
Zugewiesener Wert
 gültiger 201
Zugriff 664
 kaskadierter 177

Zugriffsoperator 102, 177
Zündholz 99
 aktivieren 99
Zusatzeigenschaft 447
Zusatzselektionsbild 434
Zustand 138
 aktiver 88, 138
 ändern 146
 inaktiver 138
 neuer 138
Zustandsautomat 276
Zuweisung 201
 Feldsymbol, dynamisch 697
 mögliche 203
Zuweisungslabor 202
Zuweisungsoperator 201, 369
Zweidimensional 84
Zweischichtiges Domänenkonzept 84,
 89, 173
Zweiter Eintrag
 lesen 224
Zwischenablage 674
 schreiben 674
Zwischenspeicherbereich 674