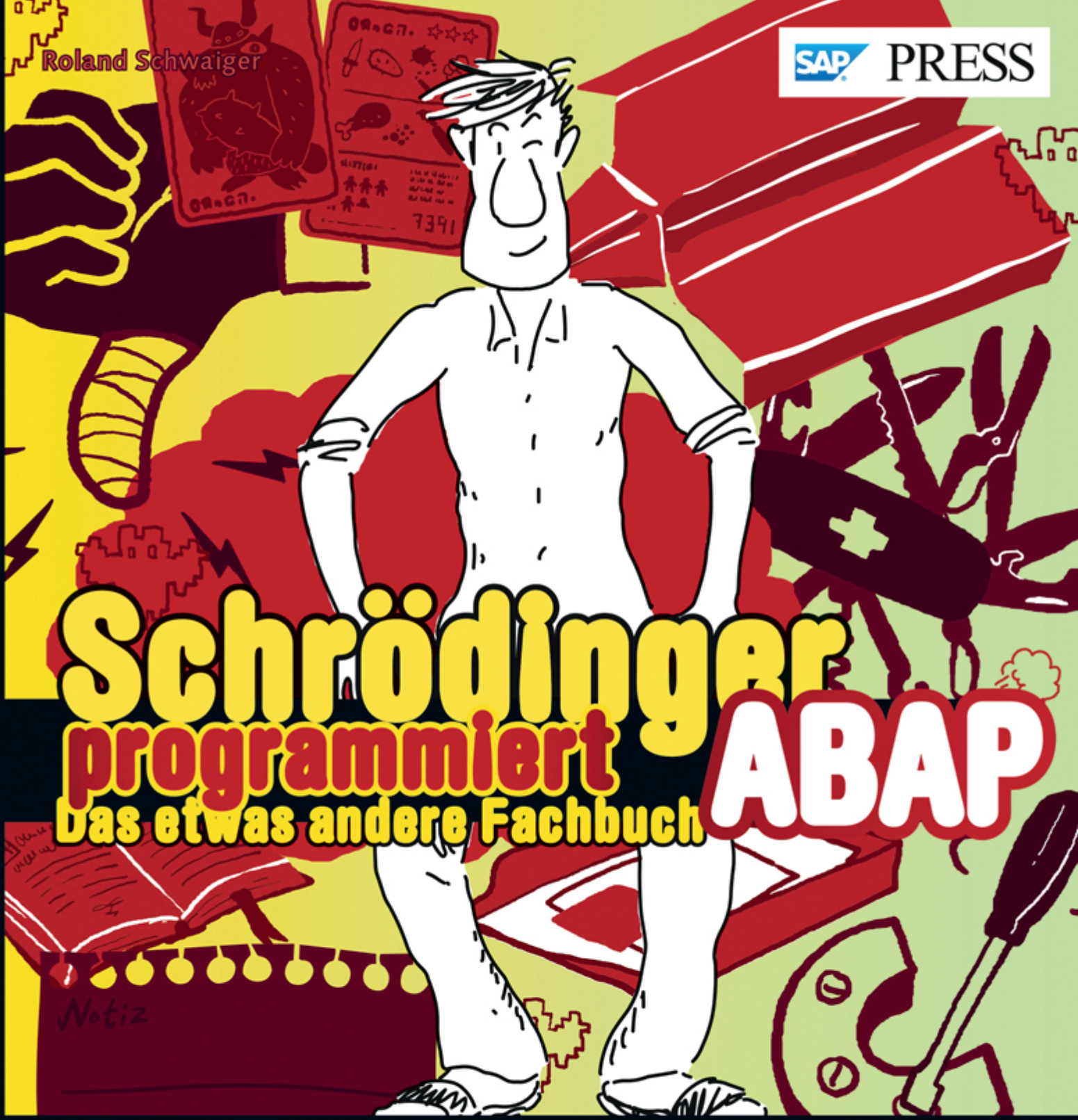
SAP PRESS

Schrödinger programmiert ABAP

Das etwas andere Fachbuch

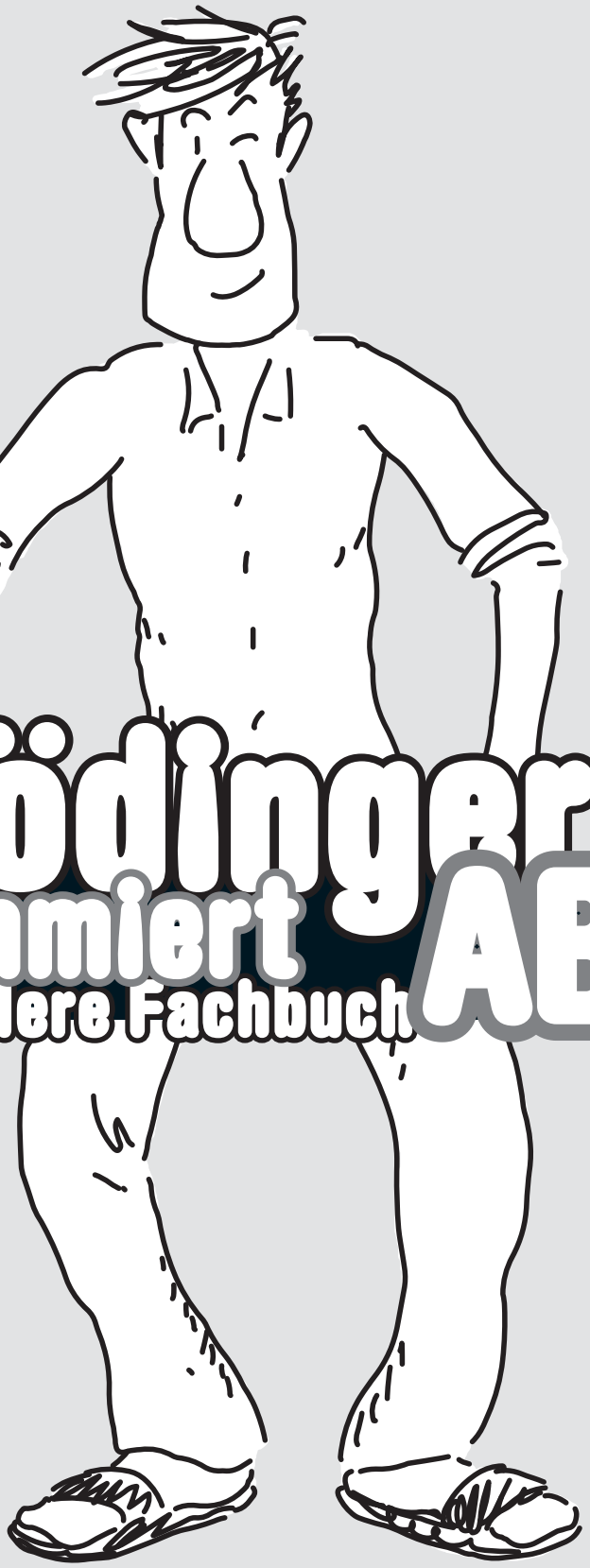
Bearbeitet von
Roland Schwaiger

Neuausgabe 2013. Taschenbuch. 735 S. Paperback
ISBN 978 3 8362 1858 0
Format (B x L): 20 x 23 cm



- Von der ersten Installation bis zur dynamischen Datenbankanwendung
- Sprachgrundlagen, Objektorientierung, Ereignisse u.v.m.: alles mit aktuellen Praxistipps
- Umwerfende Beispiele und Übungen, fantastisch illustriert

Galileo Computing



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.

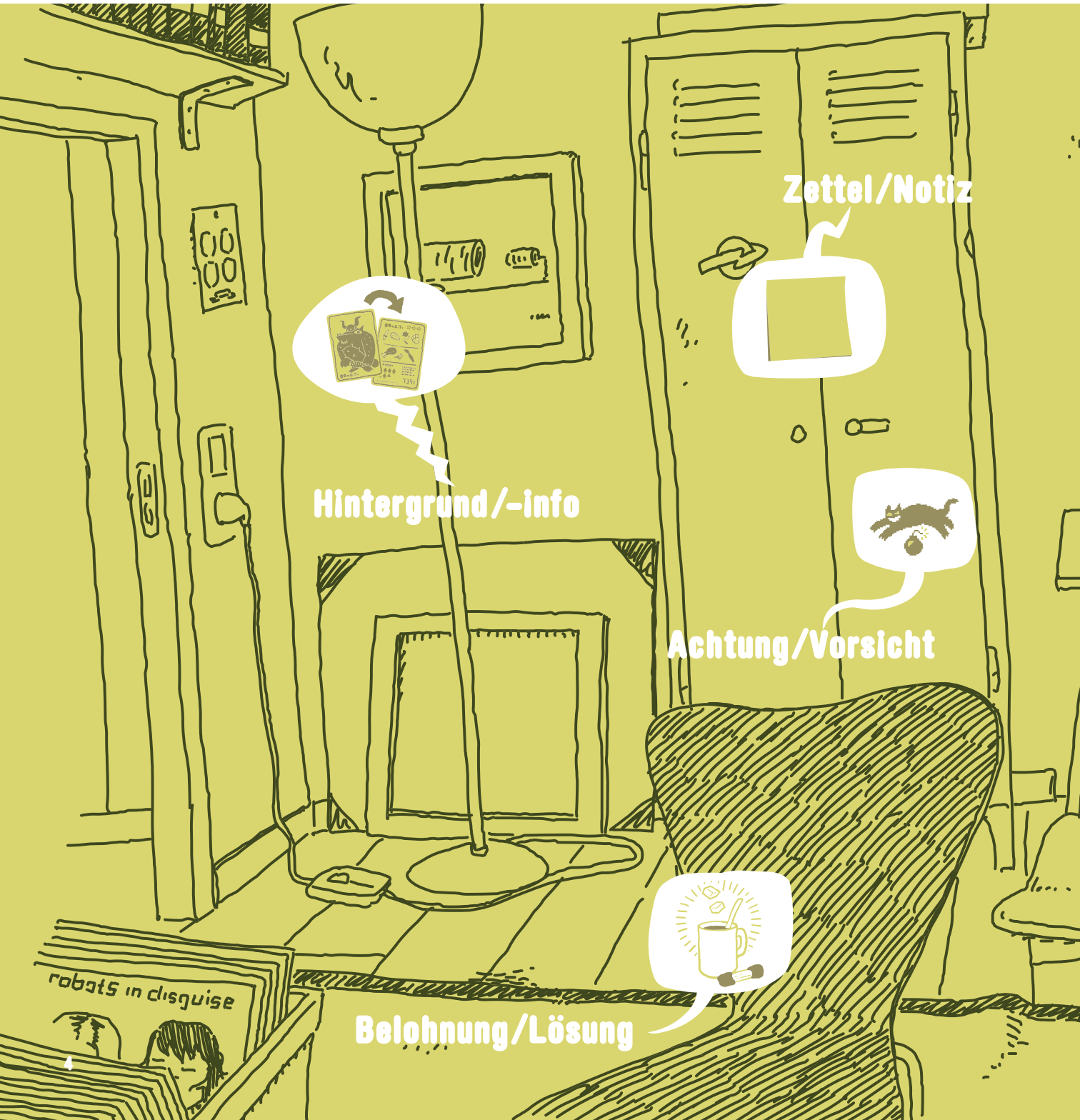
Mit Nischen, Behältern in allen Größen und belastbaren Fachböden. Wir haben keine Mühe gescheut.
Wir haben einen hervorragenden Ausbilder engagiert.
Ein Experten-Team hat die Wände behauen,
den Code eingefärbt und die nötigen Umwege eingebaut, so dass Sie bequem goutieren können.
Wobei **bequem** nicht „passiv“ bedeutet,
und **goutieren** nicht „schlafen“, aber das
wäre ja ohnehin viel zu schade.

Bitte sehr. **Es ist angerichtet.**

Viel Spaß!

Ihr Verlag

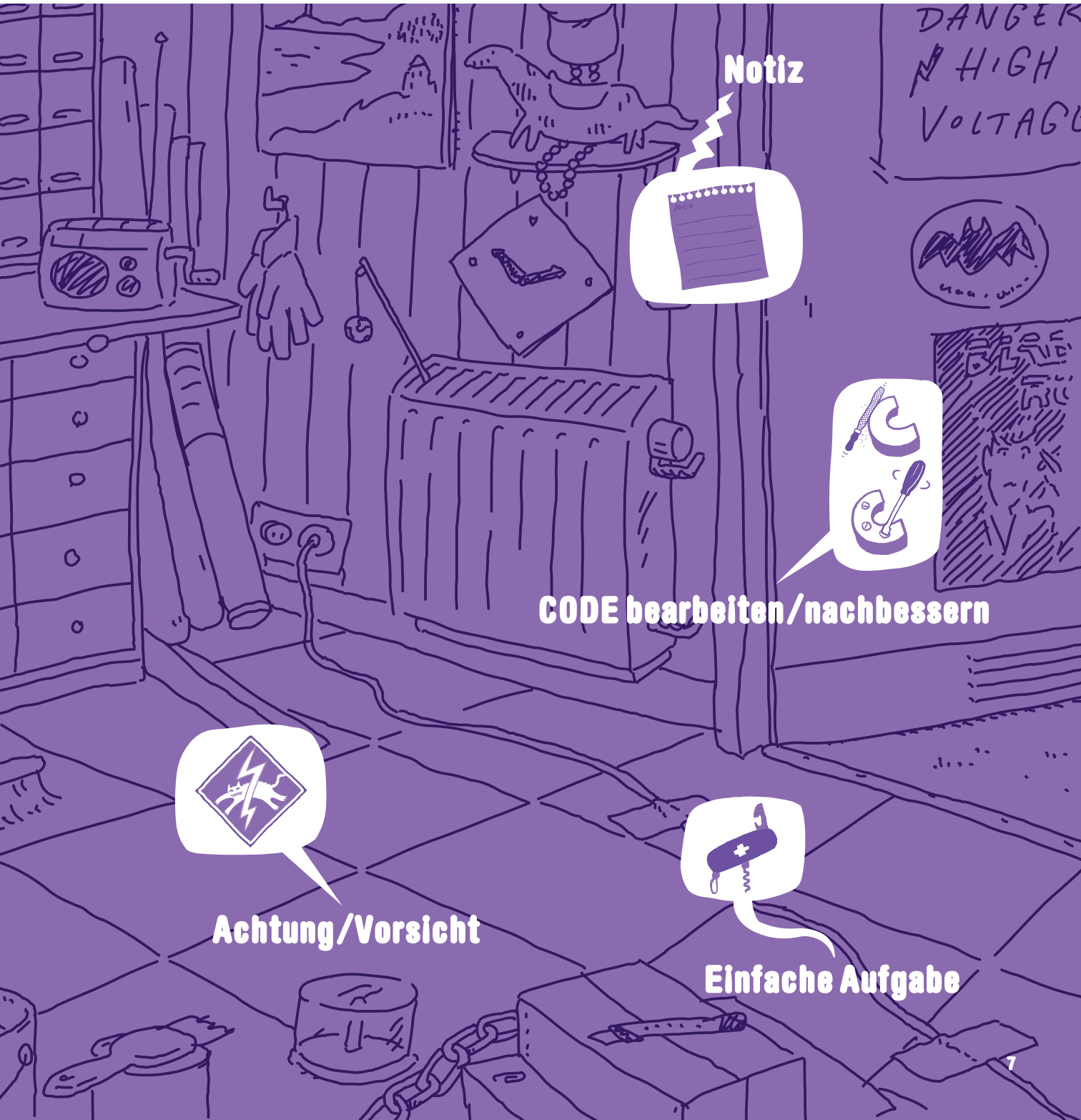
Schrödingers Büro



Die nötige Theorie, viele Hinweise und Tipps

Schrödingers Werkstatt

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



Schrödingers Wohnzimmer

Mit viel Kaffee, Übungen und den verdienten Pausen

INHALTSVERZEICHNIS

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.

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!

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.

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.

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 Pastasoße ein bestimmter Nudeltyp benötigt. Und so ist es bei der Programmierung eben auch: Für bestimmte Aufgaben werden bestimmte Typen benötigt.

Kapitel 6: Si parla ABAP? Echtes Küchenlatein

ABAP-Syntax I

Seite 197

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

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 …

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.

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?

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.

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.

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.

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 Dar-stellung 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!

Kapitel 14: Ravioli

Web-Dynpro-ABAP-Programmierung (GUI III)

Bonusseite:
http://www.sap-press.de/3024

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

Kapitel 15: Raus aus meiner Küche!

Berechtigungen

Seite 547

„Hör mal, Schwaiger Roland, zu viele Köche verderben doch bekanntlich die Nudelsoße. 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.

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.

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.

Kapitel 18: Vorratskammer in Schuss halten

DB-Daten pflegen

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!

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

Daten in Dateien, Datenablage ohne DB

„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.

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.“

Vorwort

Der Schrödinger ist bei mir eingezogen und lebt nun seit geraumer Zeit – seit Anfang 2011 – als achtes 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 Tocher 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.

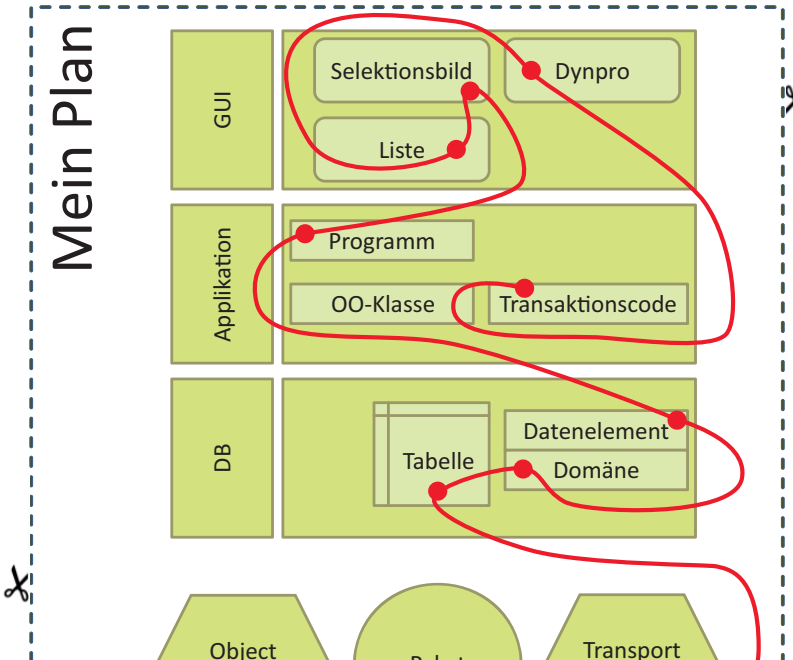
Einsteigen und anschnallen!

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!

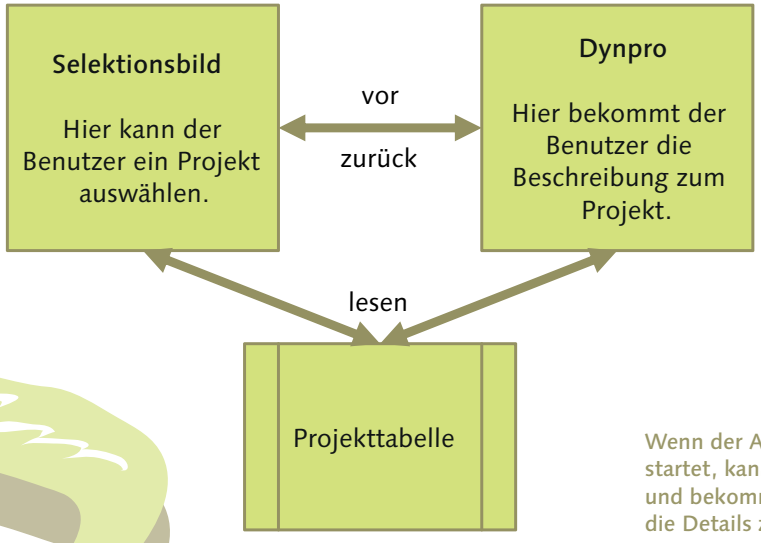


Cooler Plan, oder? Ich habe dir Markierungen zum Ausschneiden eingefügt. Schneide doch einfach den Plan aus, damit du nicht immer hin und her blättern musst.

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.

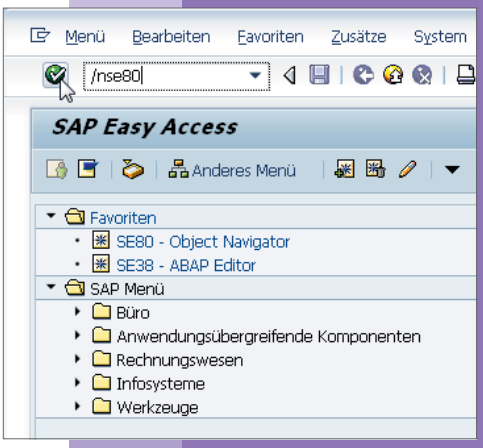
Der Anwender kann in einem **Selektionsbild** ein Projekt auswählen und Details dazu anzeigen lassen. Dafür wird ein Dynpro verwendet. Die Daten kommen natürlich, wie es sich gehört, aus einer Datenbanktabelle.
Falls dein Motor schon warmgelaufen ist, legen wir einen Le-Mans-Start hin, sprinten von der 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.

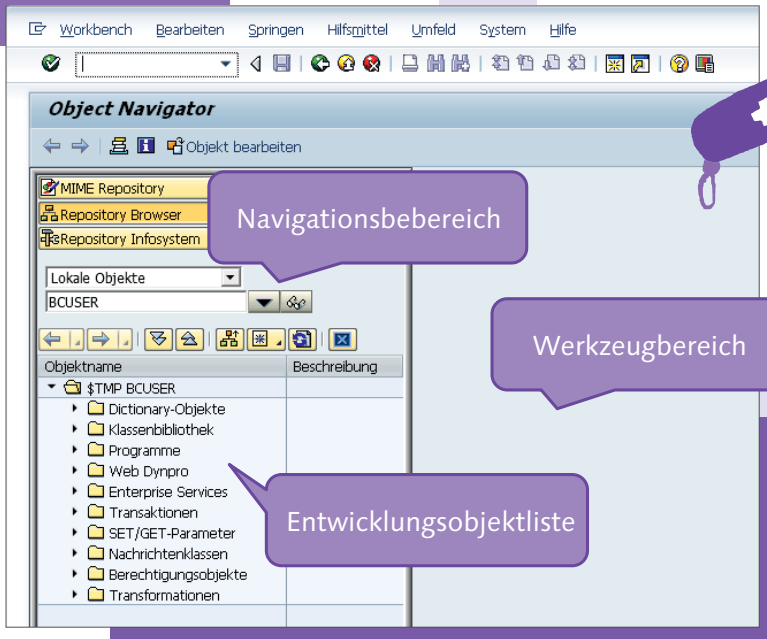
1. Tippe im Transaktionscode-Feld **`/nse80`** ein, und bestätige deine Eingabe.

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 **Werkzeugbereich** 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 wiederfindest.

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

Entwicklung organisieren – Systemlandschaft, Änderungsauftrag

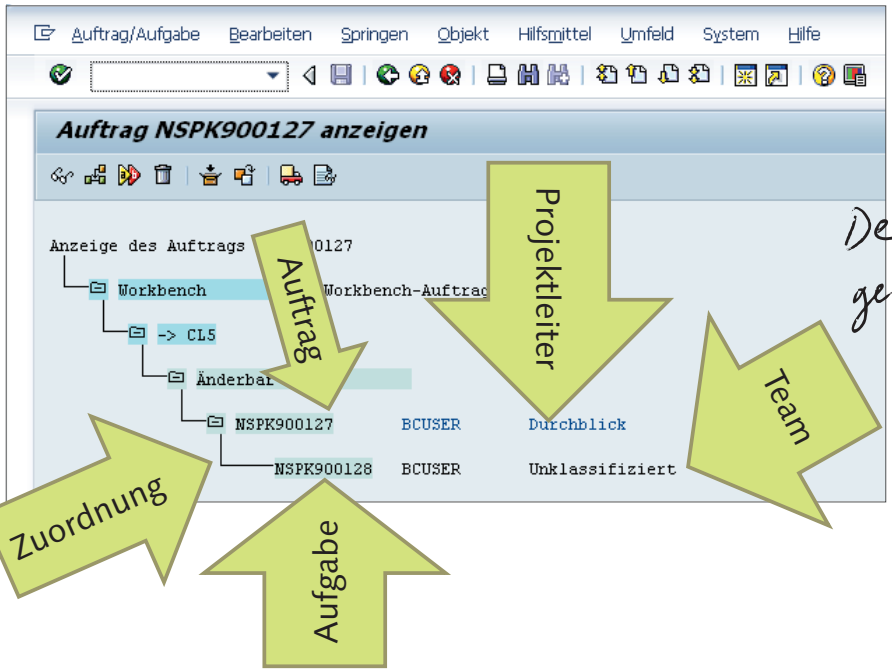
SAP empfiehlt dir und somit seinen Kunden eine **Drei-System-Landschaft**. Ein System für die **Entwicklung** – in dem werden wir uns bewegen –, ein System für die **Qualitätssicherung**, und eines für die **Produktion**, auch als **Produktivsystem** bezeichnet. Irgendwo müssen die Anwender deiner formidablen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

Bauen –
Testen –
Verwenden.

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.

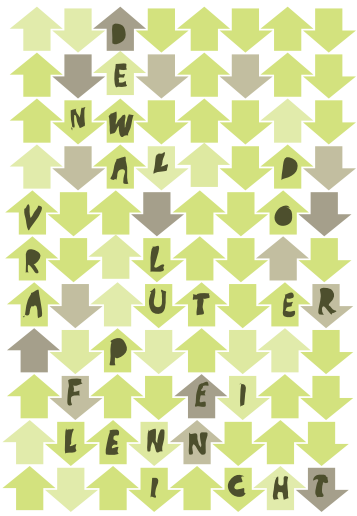
Falls dir das mit dem Engerl zu kitschig ist, kann ich dir auch eine eher formale, nüchterne Darstellung eines Änderungsauftrags (auch bekannt als **Workbench-Auftrag**) anbieten.



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.

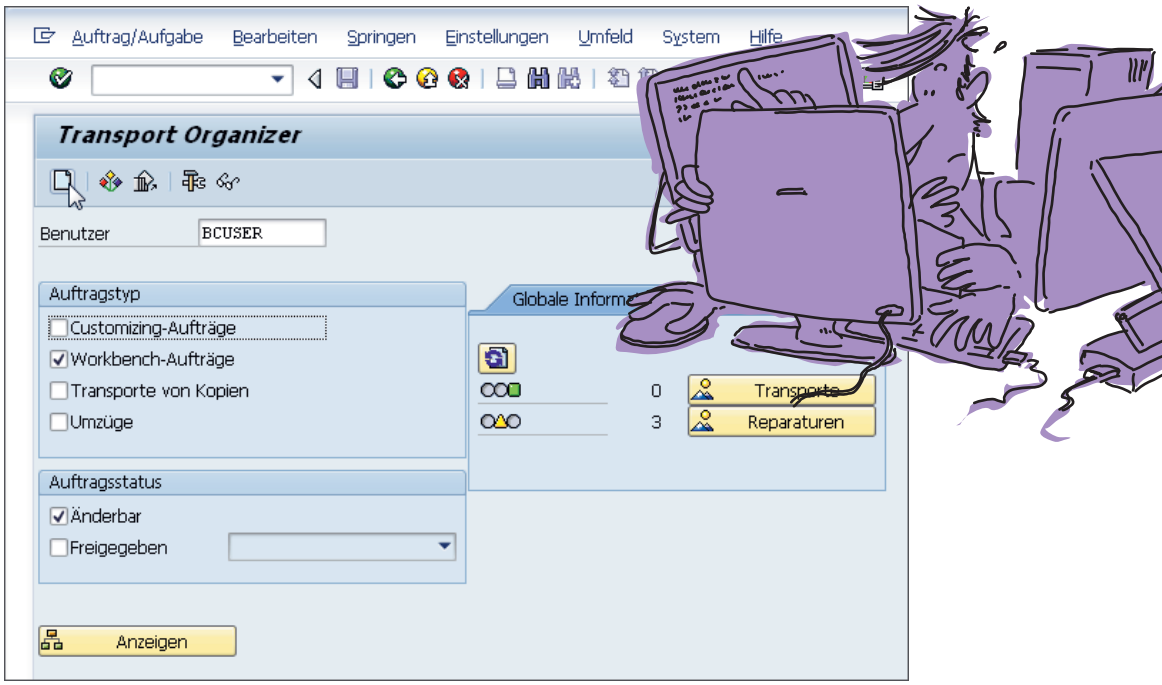


Ihr neuer Auftrag lautet: Auftragsanlage

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

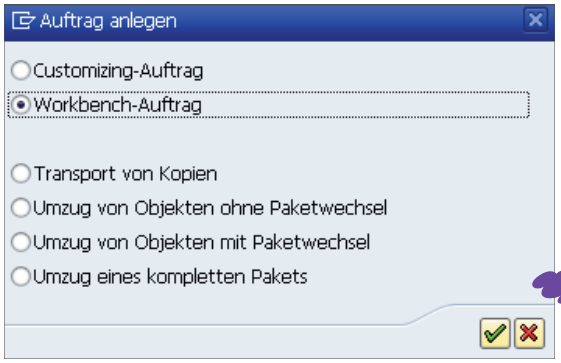
1. Ruf die Transaktion SE09 auf, am besten mit /ose09. Was geschieht beim /o nochmal?

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



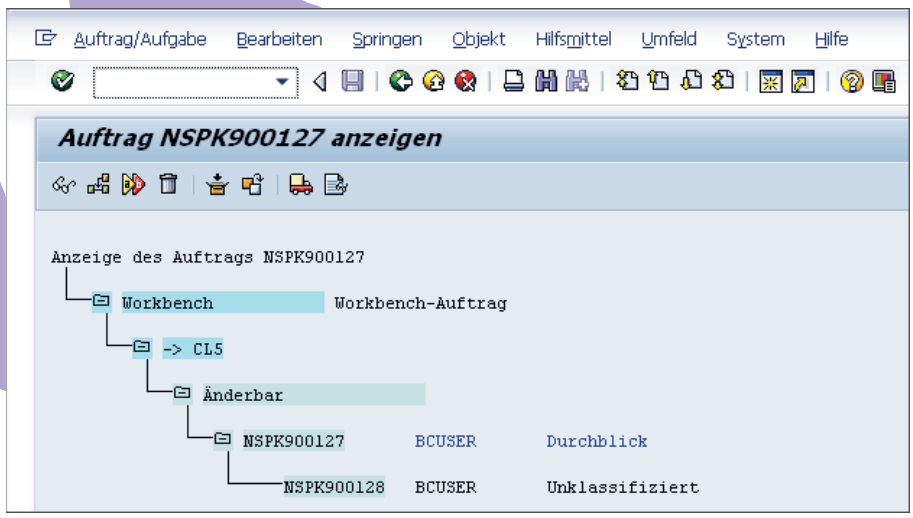
Dein erster Auftrag, den du als Projektleiter anlegst. Die Anlegen-Drucktaste (F6) macht den Job.

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**!

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**.

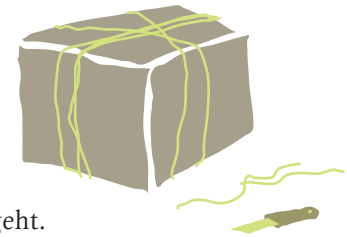
- **Standardpakete** („kein Hauptpaket") nehmen die Entwicklungsobjekte auf.
- Standardpakete können in **Hauptpakete** geschachtelt und somit gruppiert werden.
- Hauptpakete wiederum können **Strukturpaketen** zugeordnet werden.
- **Strukturpakete** werden bei der Strukturierung großer Softwareprojekte 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 Einspielens neuer SAP-Entwicklungsobjekte nicht überschrieben.

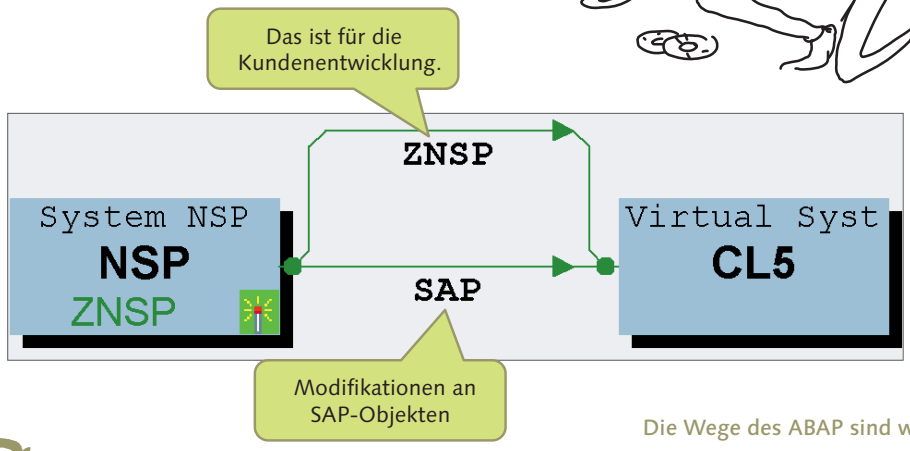
Safety first!

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 Banen - Testen - Verwenden 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.

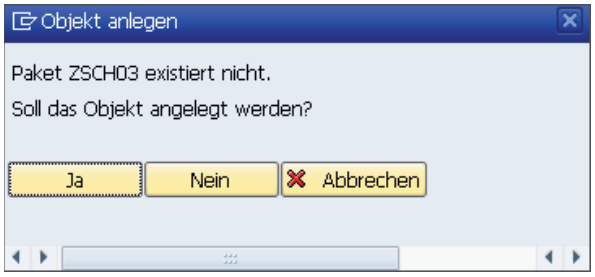
1. Wechsle in den Object Navigator. Das sollte ziemlich einfach sein, da der Modus mit dem Object Navigator immer noch vorhanden sein sollte. Wenn nicht: auch egal, einfach nochmal **SE80** aufrufen.

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. Den wünsche ich mir zu meinem Geburtstag.

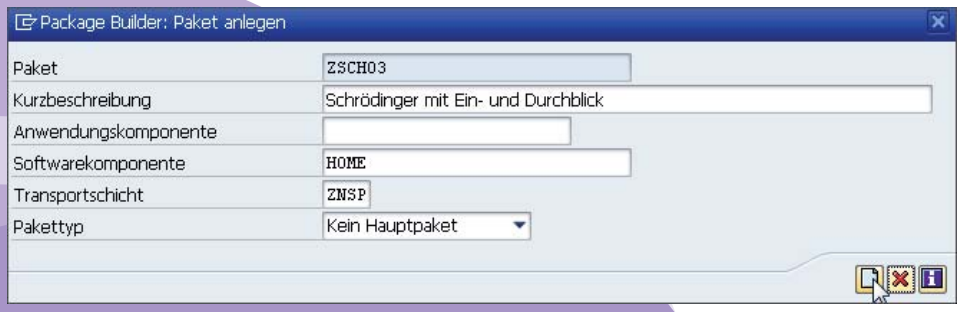
Du willst also ein Paket anlegen. Bravo, und auch noch so ein schöner Name.

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**.



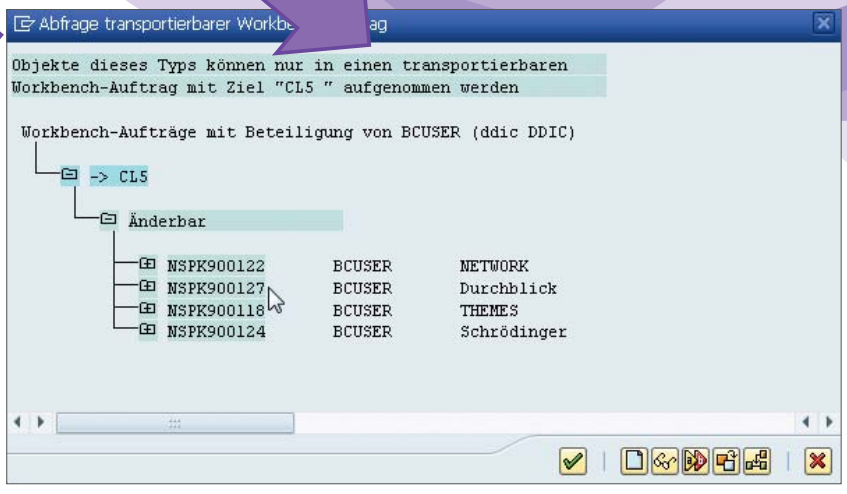
Ja, ich will anlegen, bitte.

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).



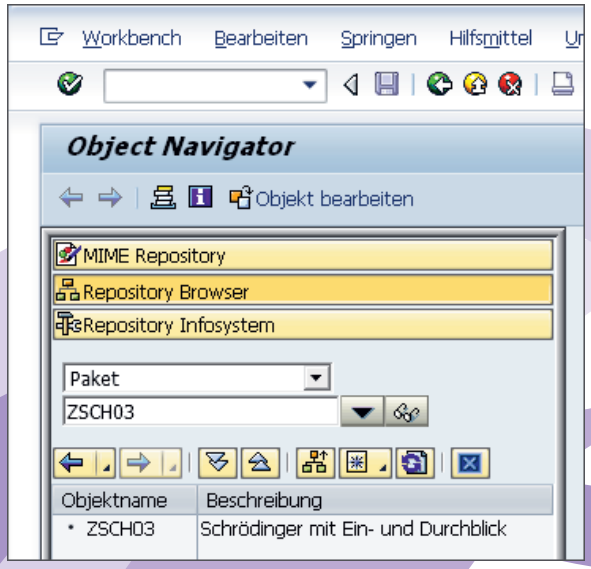
Ein neues Standardpaket (kein Hauptpaket), das keiner Anwendungskomponente und der Softwarekomponente HOME zugeordnet wird.

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.



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!**

[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.

Paaaaaause, bitte.

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 ' _]
[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.

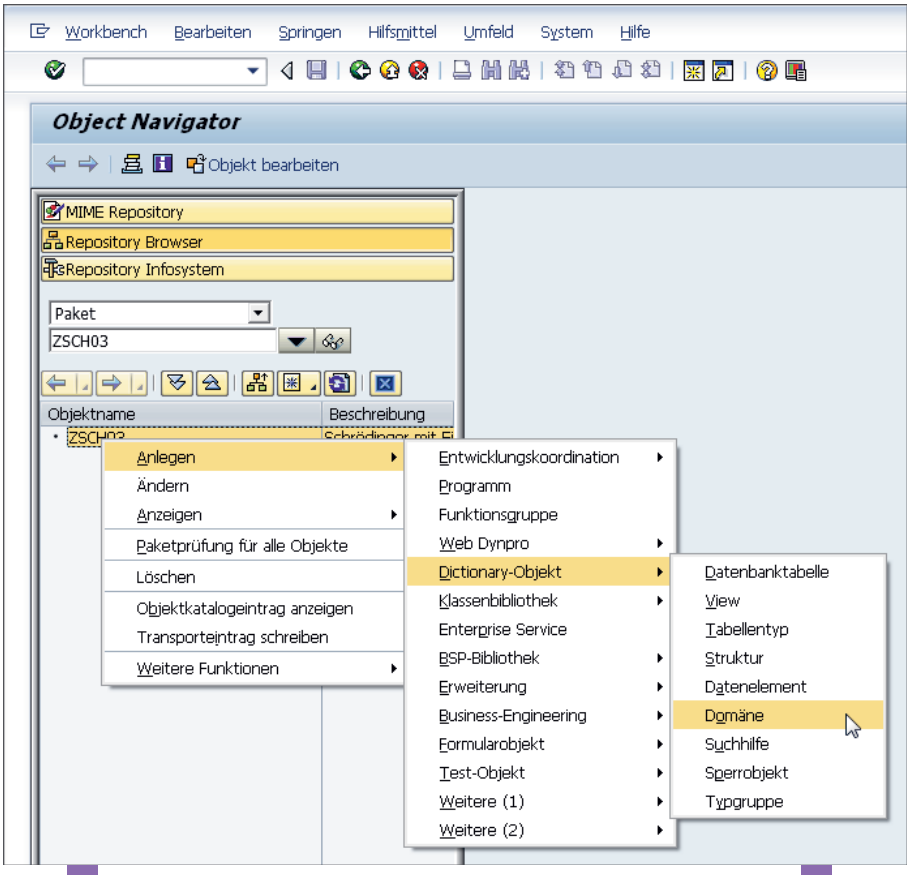
Sicher nicht. Jetzt ist endgültig Pause.

Okay, dann wirf ein paar Bälle.
Ich bereite in der Zwischenzeit deine nächste Gehirnnahrung vor.

Technisches Fundament eines Typs – Domäne anlegen

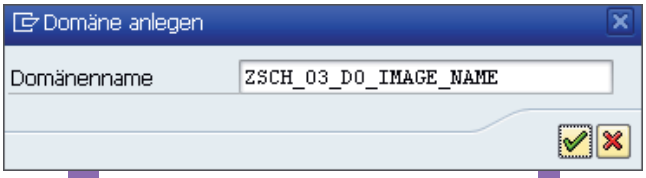
Wir werden uns jetzt eine Domäne für die Tabellenspalte zum **Bildnamen** anlegen.

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.



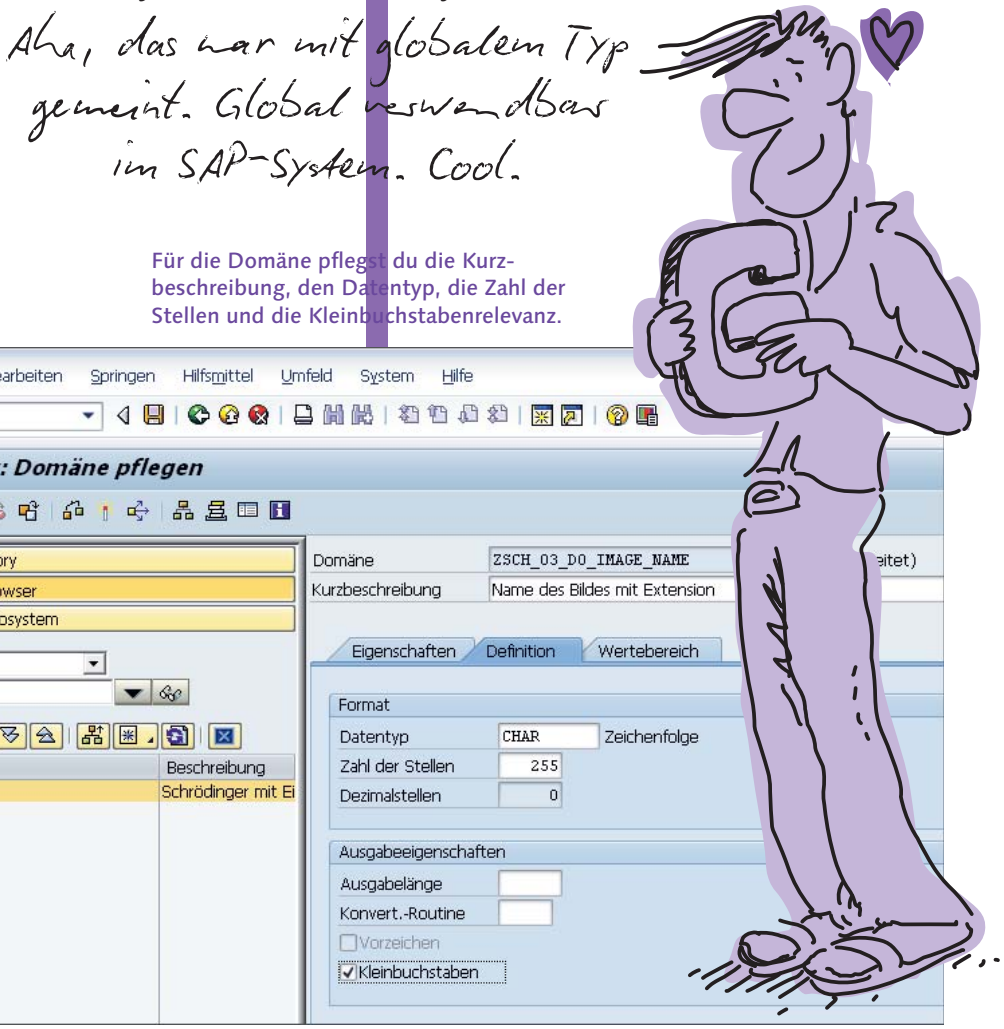
Das Anlegen einer Domäne mithilfe des Kontextmenüs des Pakets.

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.



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!

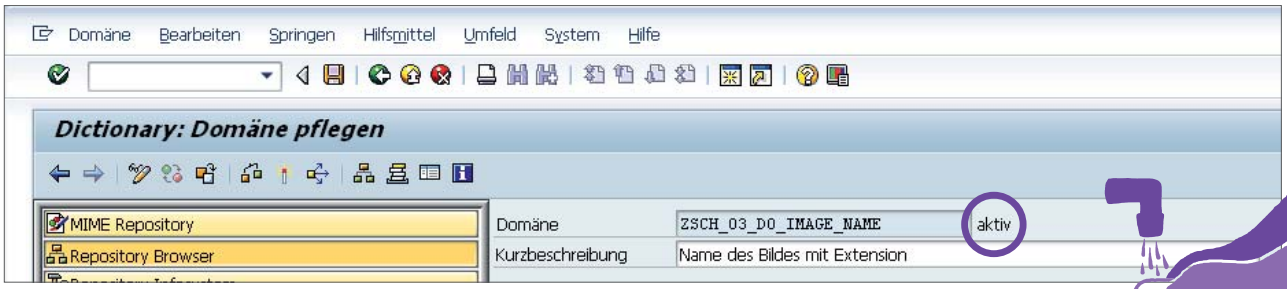
Beim Sichern erscheint ein Dialog mit dem Titel **Objektkatalogeintrag anlegen**, und da wird das Paket ZSCH03 vorgeschlagen. Ist das okay, dass ich den Sichern-Button angeklickt habe?

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!]

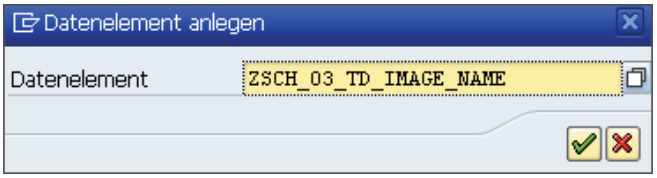
So schnell kann es gehen. Du hast deine erste Domäne angelegt. Perfekt. Als Nächstes kommt die Verwendung in einem Datenelement. SAP sagt dazu auch die …ntische Ebene.

Romantische? Scherzerl: Natürlich semantische!

Lege die Bedeutung an – Datenelement anlegen

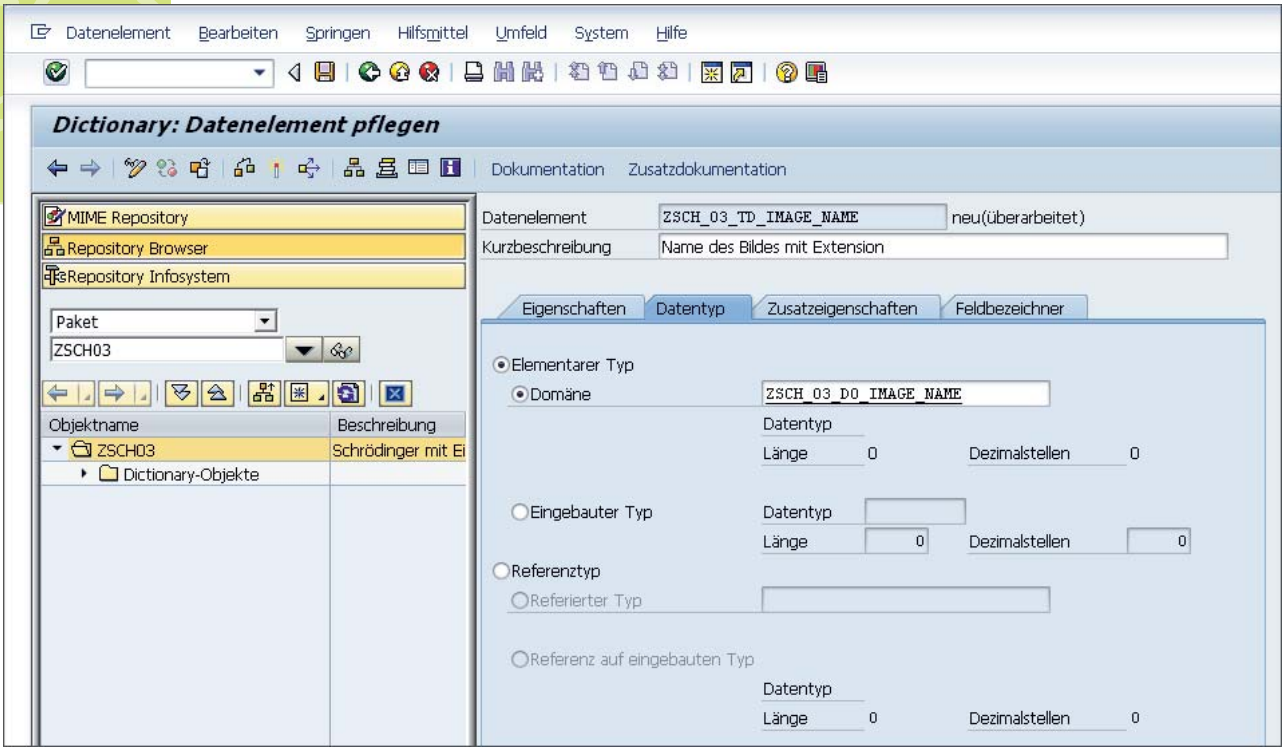
Wir benötigen ein Datenelement für die Typisierung der Tabellenspalte **Bildnamen**. Also produzieren wir es:

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.

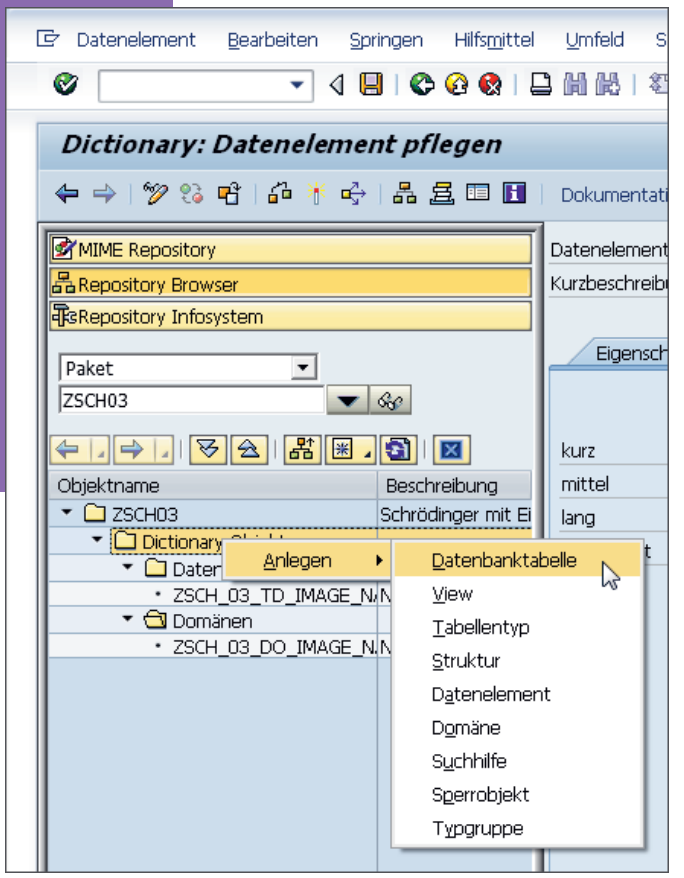
Kühlschrankdesign – DB-Tabelle anlegen

Alles ist bereit und vorbereitet. Lasst uns froho uhund 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.)

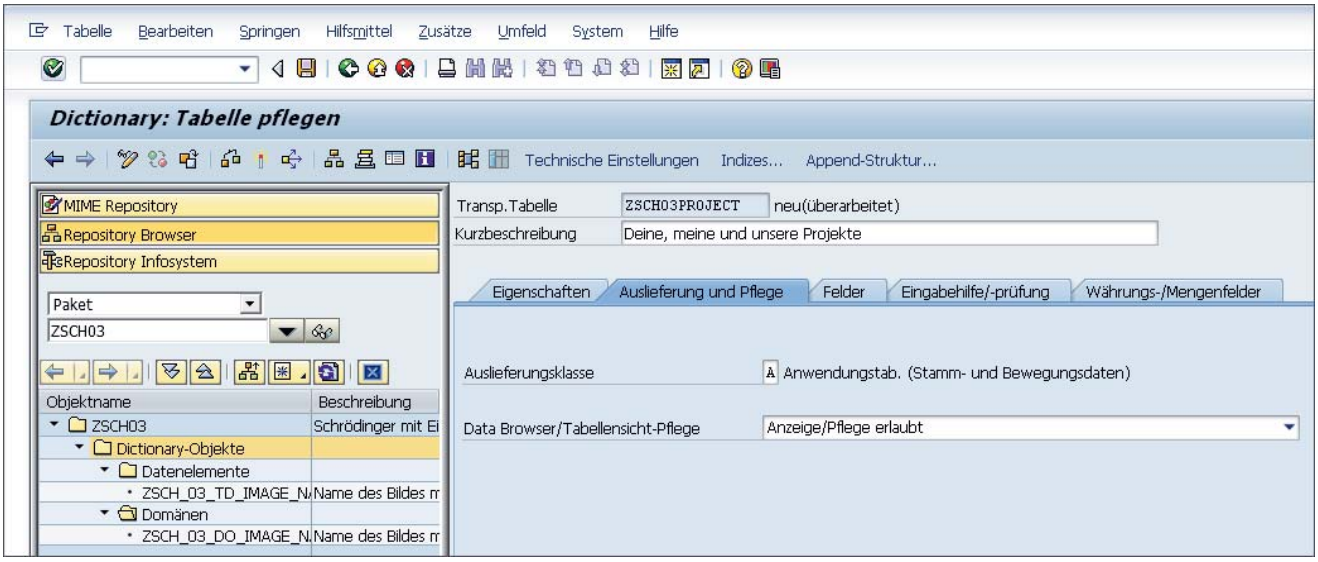


Ein alternativer Pfad zur Anlage eines Dictionary-Objektes. Jetzt geht es um Datenbanktabellen.

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.

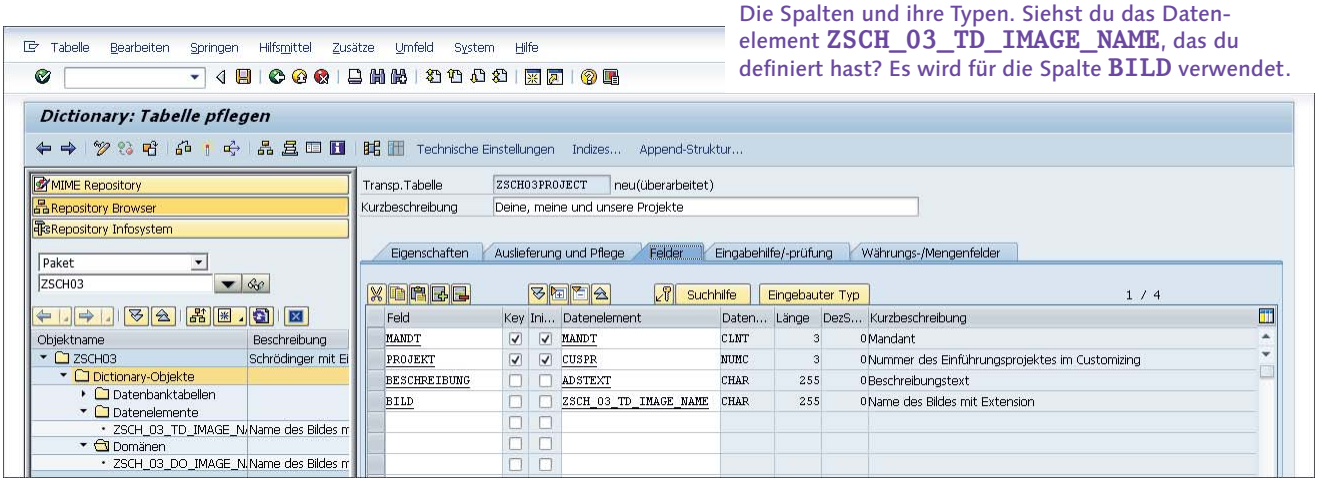


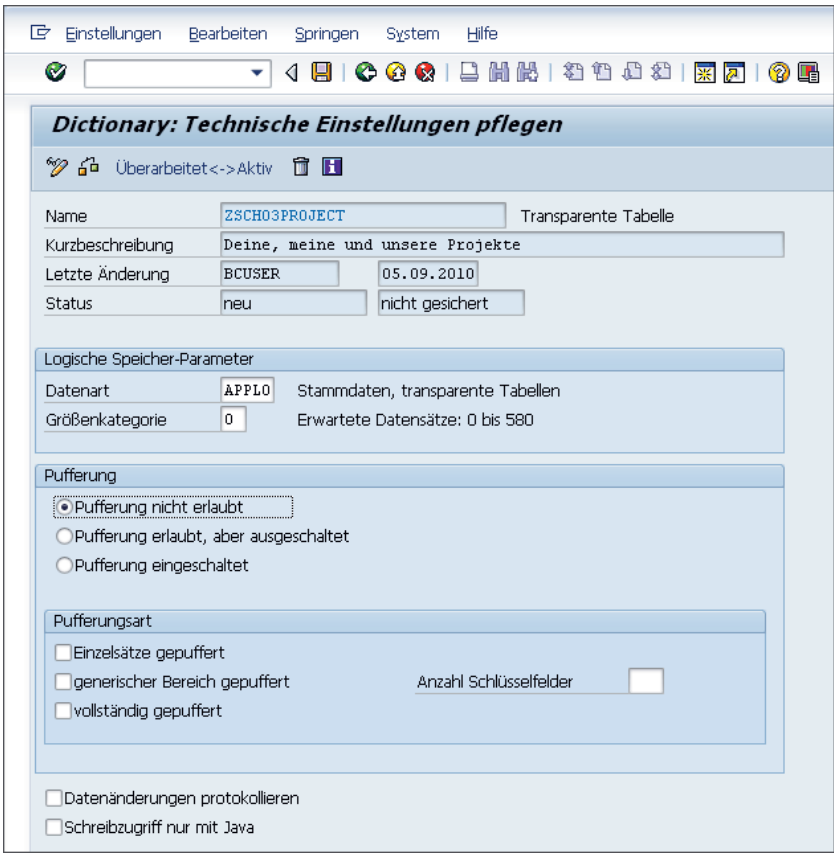
Welche Art von Daten und manueller Pflege, das sind die Fragen!

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.





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

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.

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.

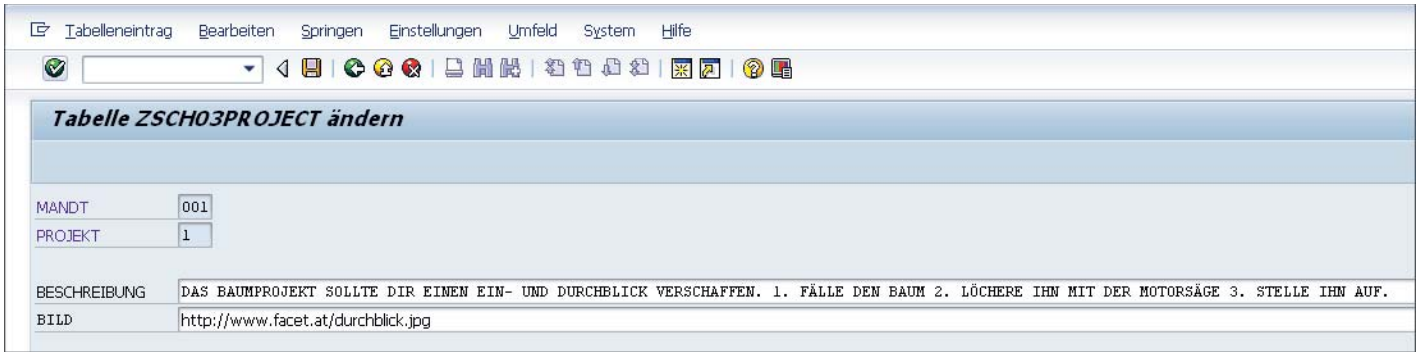
8. Nachdem du die **Übernehmen**-Taste gedrückt hast kommt wieder deine Lieblingssequenz: Sichern und aktivieren. Fertig!

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.



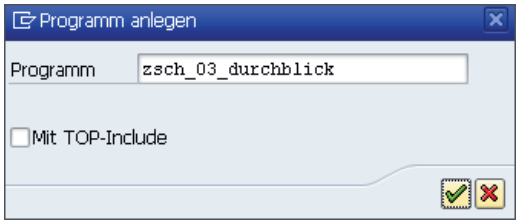
Ein Eintrag in deiner Tabelle. Das Lochbaumprojekt.

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.

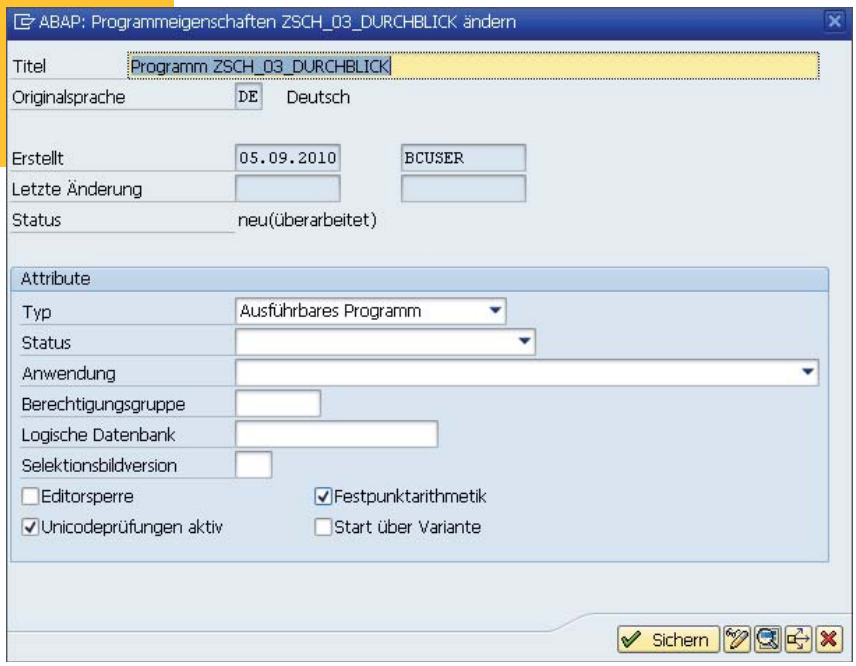
Daten verarbeiten – Programm anlegen

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.



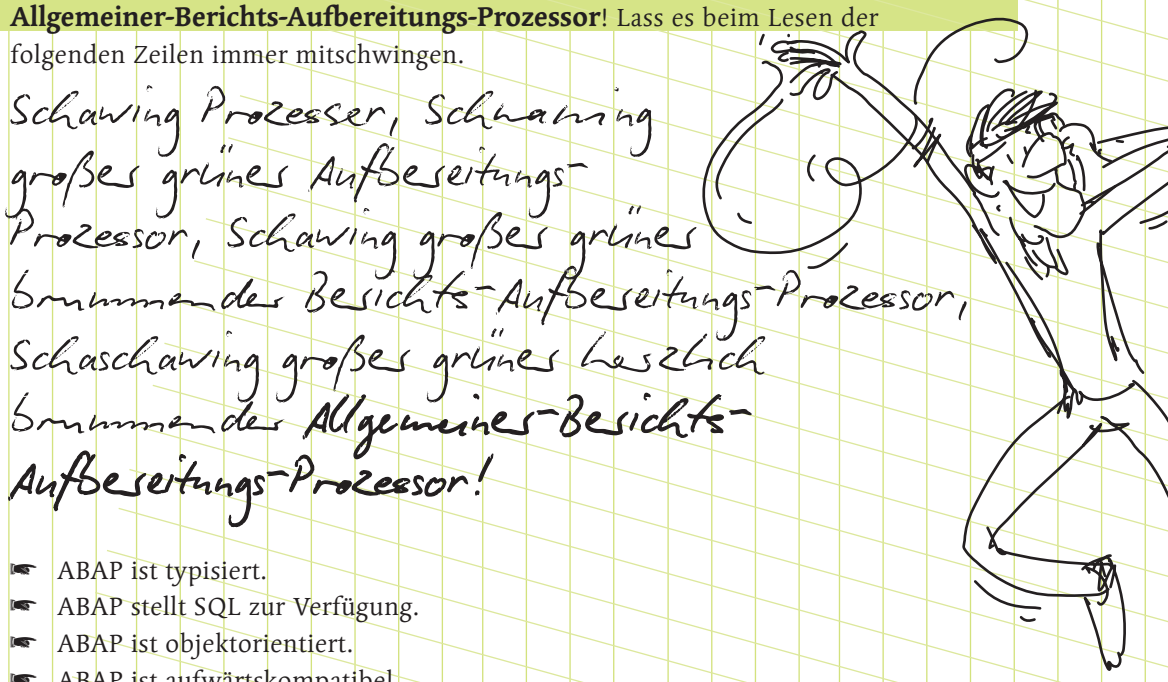
Was kann man denn da alles einstellen? Wichtig für jetzt ist nur, dass du ein **Ausführbares Programm** anlegst.

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.

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.



- 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'.
```

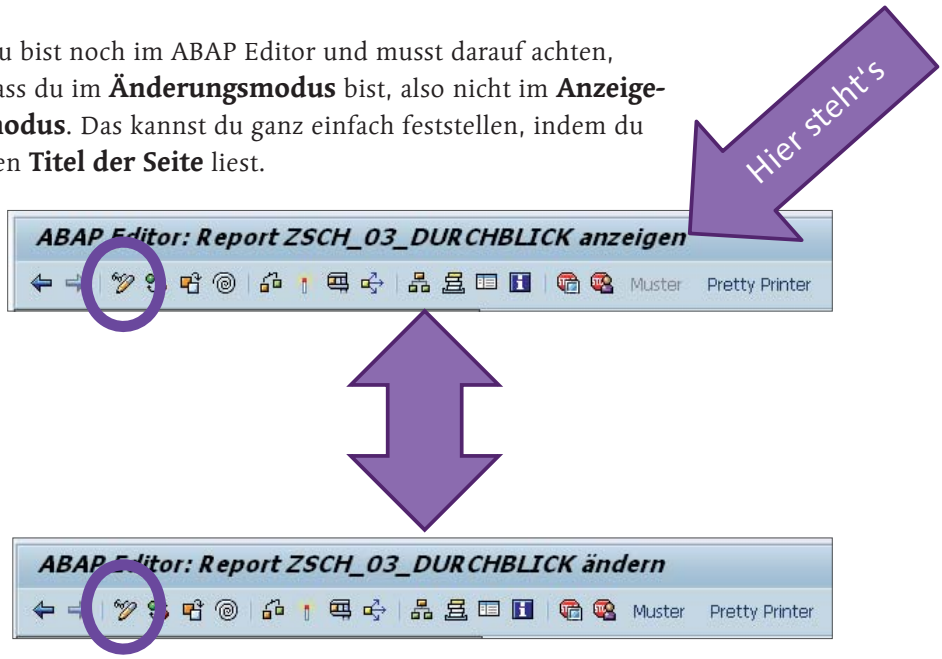
Einstieg in die **Modularisierung** the easy way: In ABAP kannst du ein ausführbares Programm (den Programmtyp verwenden wir für unser Beispiel) mithilfe von **Ereignisschlüsselwörtern** untergliedern. Zu bestimmten **Zeitpunkten** werden die so markierten Blöcke automatisch vom **Laufzeitsystem** ausgeführt, 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.



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.

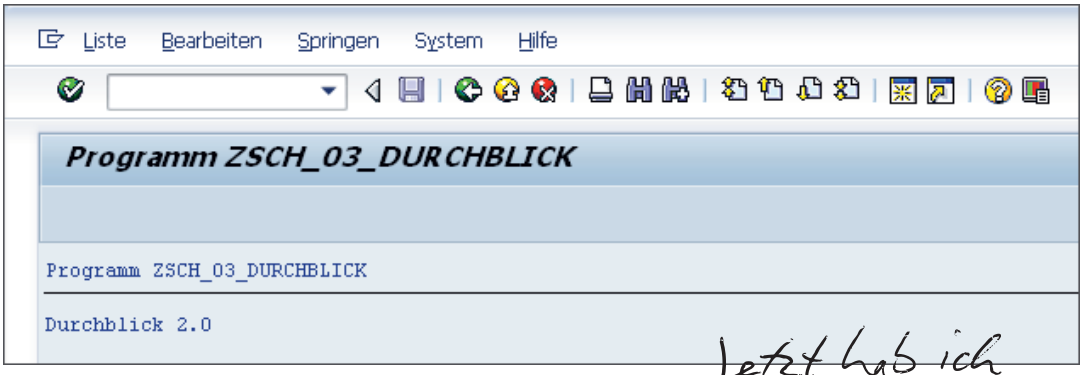
3. Aktivieren mit der **Aktivieren**-Drucktaste (Strg +F 3) (die mit dem Zündholz) in der Drucktastenleiste.

4. Testen mit der **Direkt**-Drucktaste (F8) (die mit dem Schublehresymbol, könnte aber auch eine Schraubzwinge sein).

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

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'.
```

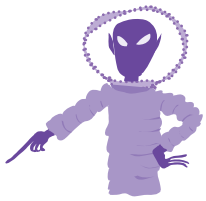


Wunderschöne Liste mit deiner Ausgabe. Willkommen im Club.

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 **Projekttabelle** 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

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.

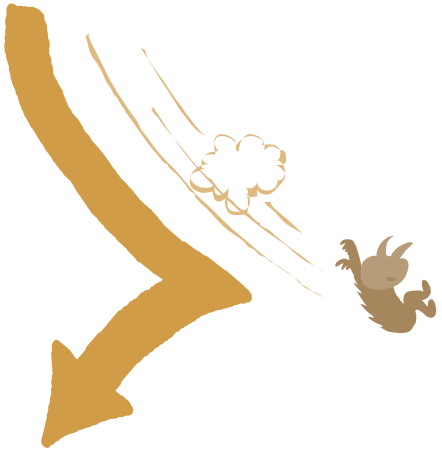
Ein Eintrag aus der Datenbanktabelle in deiner Liste. Damit kommunizierst du vom Programm aus mit der Datenbank.

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 vorhin.

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.
```

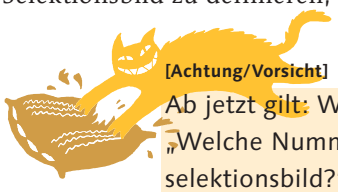
Kurzer Fokus: Prüfe nach dem **SELECT SINGLE**, ob ein Wert in der Datenbanktabelle gefunden wurde. Verwende dazu **IF, ELSE, ENDIF**, **=**, **sy-subrc**, den Wert **0**. Dieses Mal habe ich einen Lückentext für dich.

```
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:

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

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

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

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

*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 **Wertehilfe**. Das kommt dann alles ausführlich in Kapitel 12.

[Einfache Aufgabe]

It's exercise time!

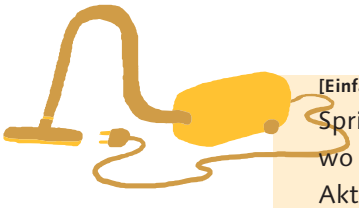
Implementiere im Programm, direkt unter der REPORT-Zeile, den Parameter für die Eingabe des Projekts, und verwende den Zusatz OBLIGATORY am Ende.

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

Nö. Etwas 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]

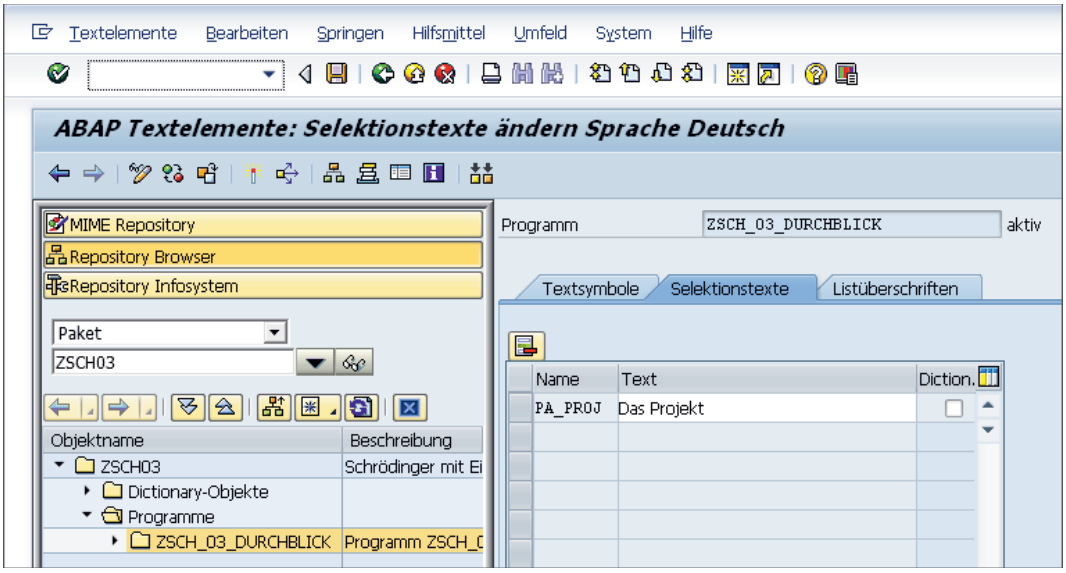
Springe zu den Selektionstexten, tippe direkt neben dem Parameter PA_PROJ, dort, wo das Fragezeichen steht, den Text „Das Projekt“ ein, und aktiviere den Text mit dem Aktivieren-Button.
Wechsle danach wieder zurück (F3), und starte das Programm.

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.
```

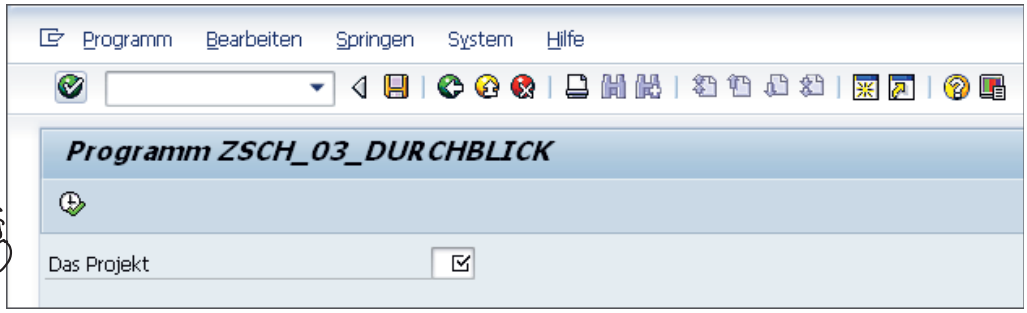
*1 Da ist der Parameter aber schön zwischen **REPORT** und **DATA** eingeklemmt. Der neue Teil ist der Zusatz **OBLIGATORY**. Der bewirkt, dass der Anwender einen Wert eingeben **MUSS**. Falls der Anwender trotzdem das Programm ohne starten möchte, wird er mit Fehlermeldungen überflutet.

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 West ich eingebe und das Programm mit dem grünen Häkchen (F8) starte, ich bekomme immer denselben Datensatz. Super Selektionsbild

Ahhh

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. Eeeeeeeh. 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.

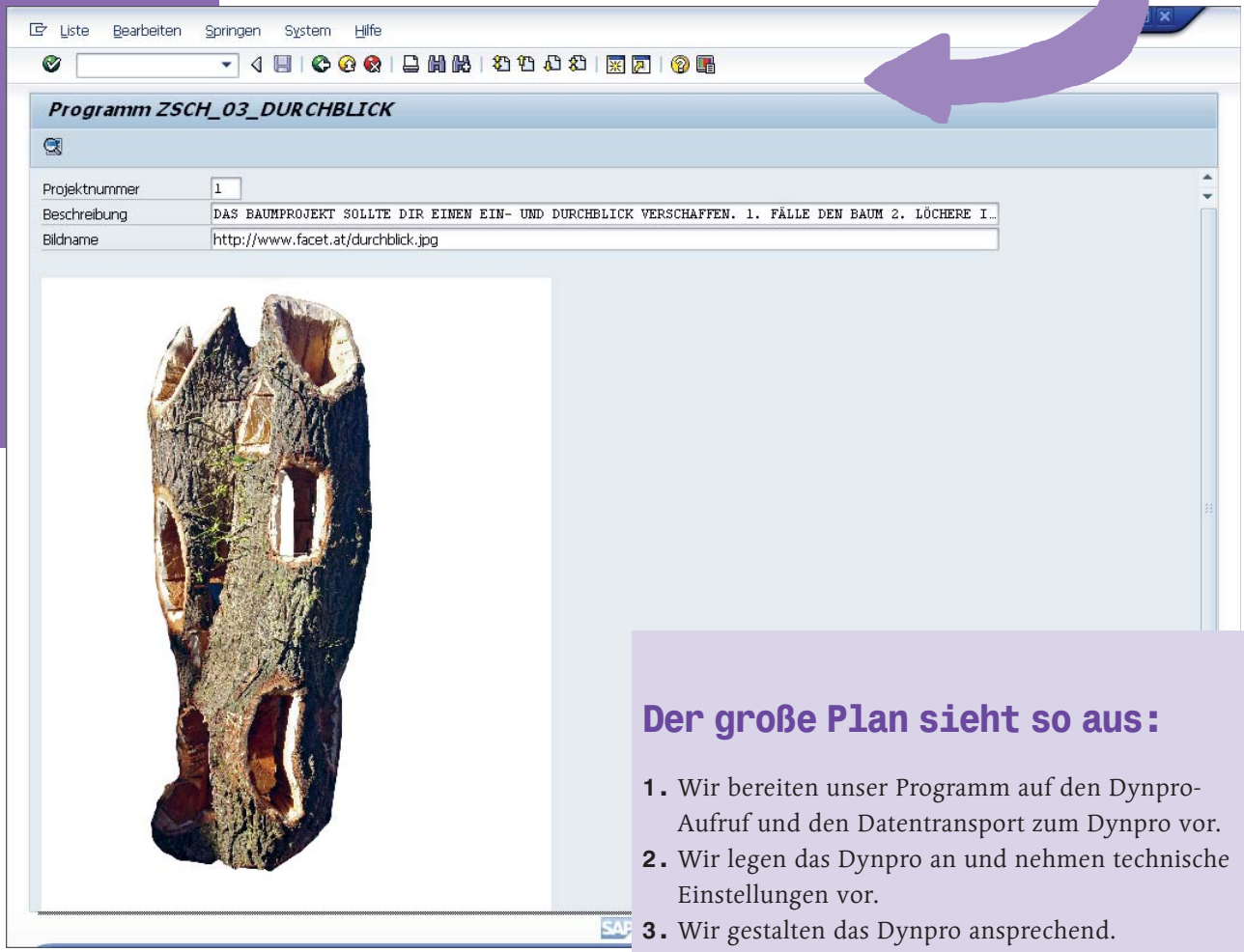
Eine etwas komplexere Form der Eingabe wird mit **SELECT-OPTIONS** definiert. Dabei werden Intervalle von Werten für die Eingabe angeboten. Das ist aber für den Augenblick zu kompliziert und wird daher auf Kapitel 12 verschoben.

[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:**



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 Datenbanktabelle 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 …?

9000, weil das die Nummer des Standardselektionsbildes ist. Jetzt bin ich aber selber etwas baff. Also diese modernen Energydrinks …, ob das noch legal ist?

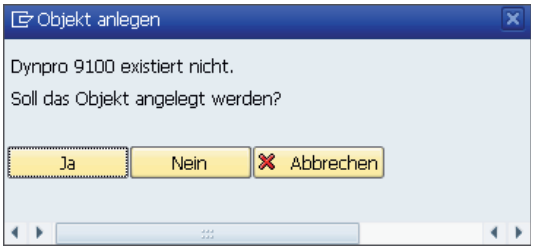
[Einfache Aufgabe]
Implementiere die besprochenen Änderungen im Programm. Sobald du den Befehl **CALL SCREEN 9100.** eingefügt hast, sicherst du und klickst doppelt auf **CALL SCREEN**. Und dann wartest du auf mich!

Dekorieren der Auslage – Dynpro-Layout

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

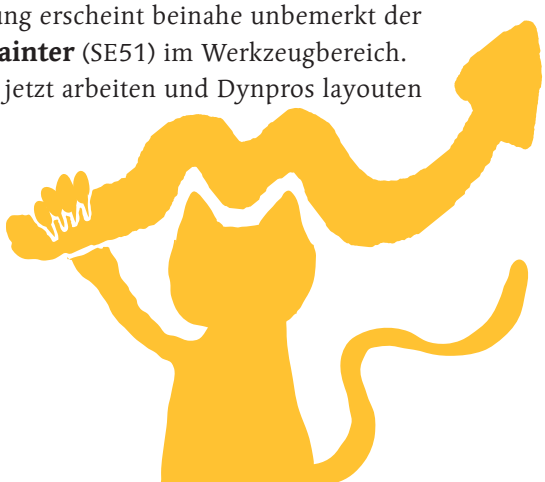
Dynpro anlegen und technische Einstellungen vornehmen

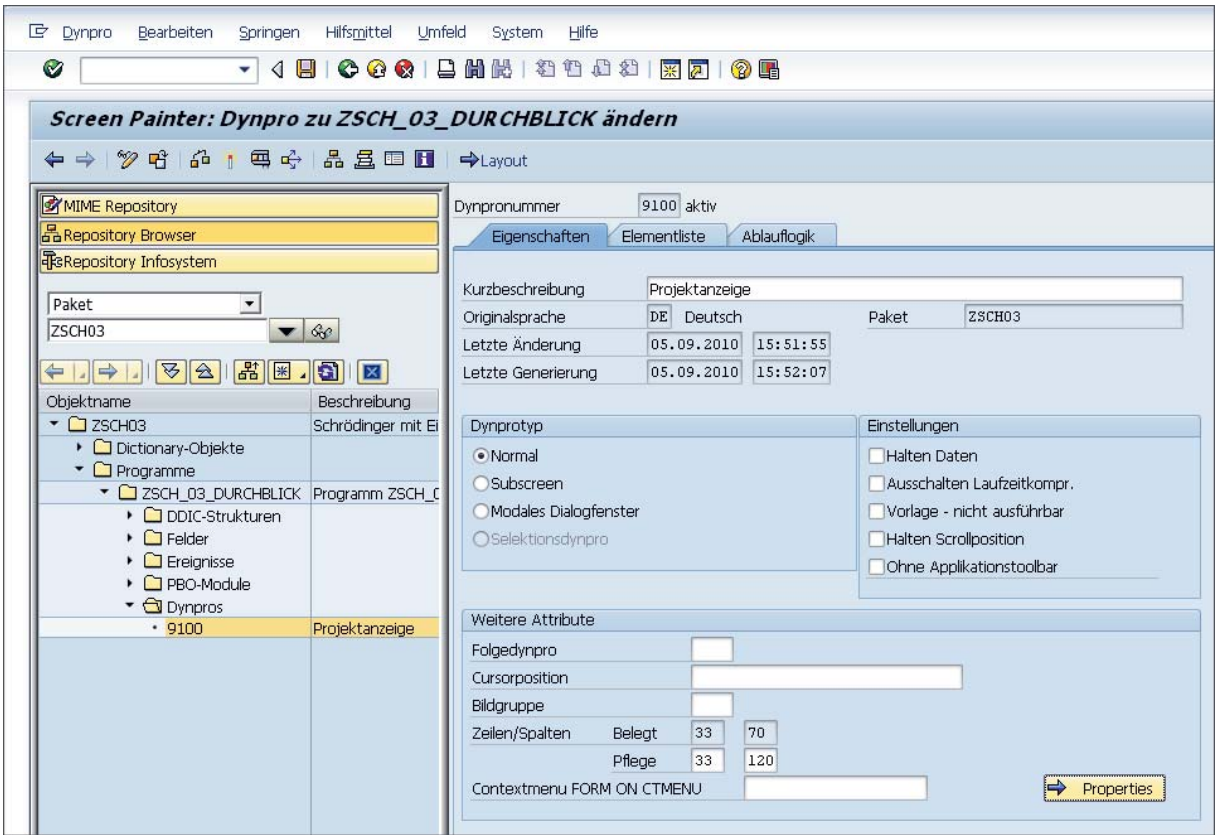
1. Du bist also schon beim Pop-up zum Anlegen des Dynpros. Das kannst du einfach mit **Ja** bestätigen, weil du ja willst, dass das Dynpro angelegt wird.



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 layouten und programmieren.





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

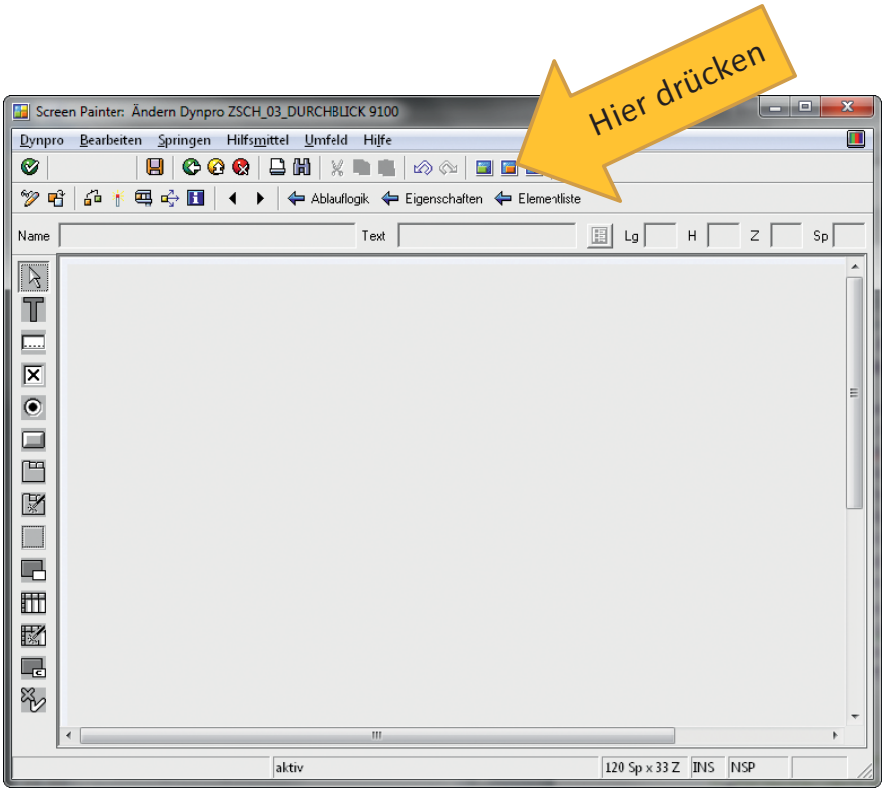
Auf dem Karteireiter **Eigenschaften** pflegst du für unser Beispiel die **Kurzbeschreibung**, die du dann in der Objektliste wiederfindest, und das **Folgedynpro**. Das steuert, wohin nach der Abarbeitung der Ablauflogik, das ist das Programm zum Dynpro, gesprungen wird. Wenn du die Nummer **9100** 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.



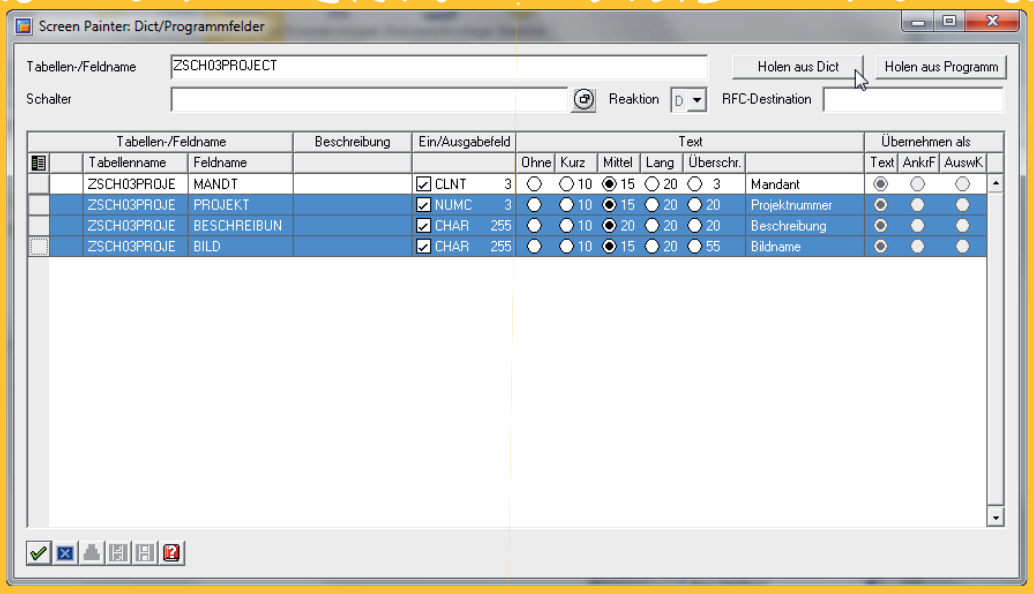
So sieht ein leeres Dynpro aus, ziemlich nackt.

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.

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?

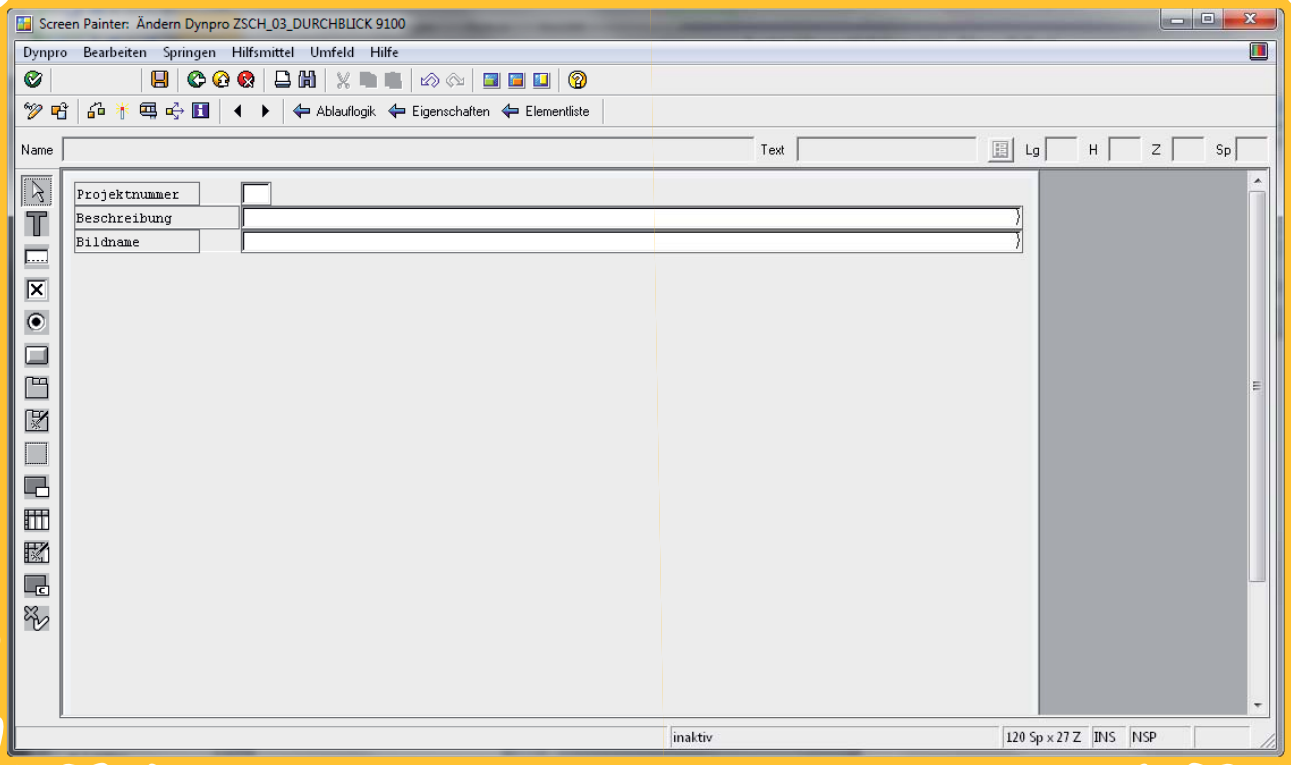
- ☐ A grüner Haken
- ☐ B blaues X
- ☐ C rotes Fragezeichen

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.

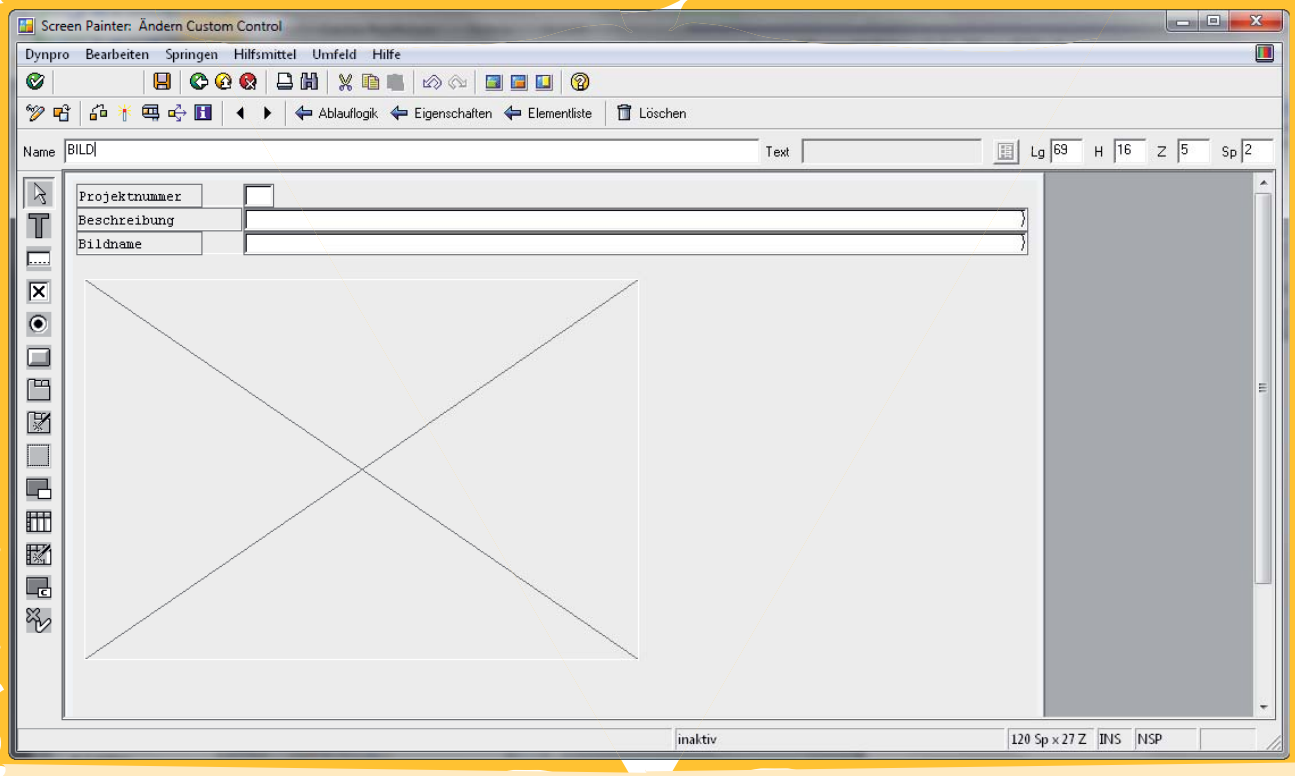
Platziere die drei Felder ganz links oben auf dem Dynpro, indem du dorthin klickst. Plumps, die Felder sind platziert, und du erhältst noch eine Warnung, die dich nicht stören sollte. Drück sie einfach weg, und denke dir: Mir egal!



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 Creuzworträtsel, 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.

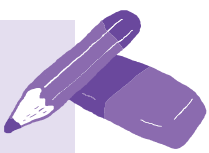
6. Wieder zurück mit der Zurück-Taste (grüner Pfeil) oder F3.

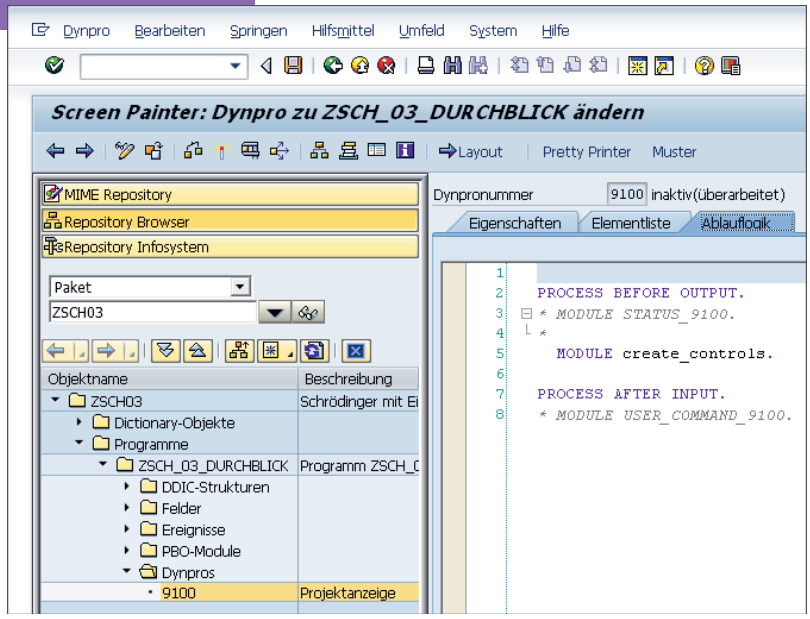
Ablauflogik ohne Ablaufdatum – Ablauflogik programmieren

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.





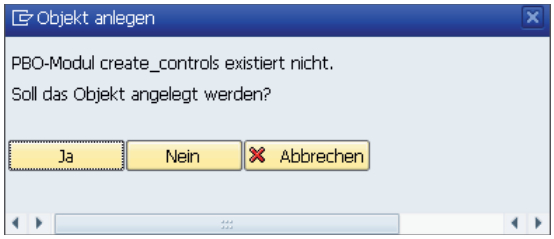
1. Wechsle bitte auf den Karteireiter **Ablauflogik** im Screen Painter.

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.

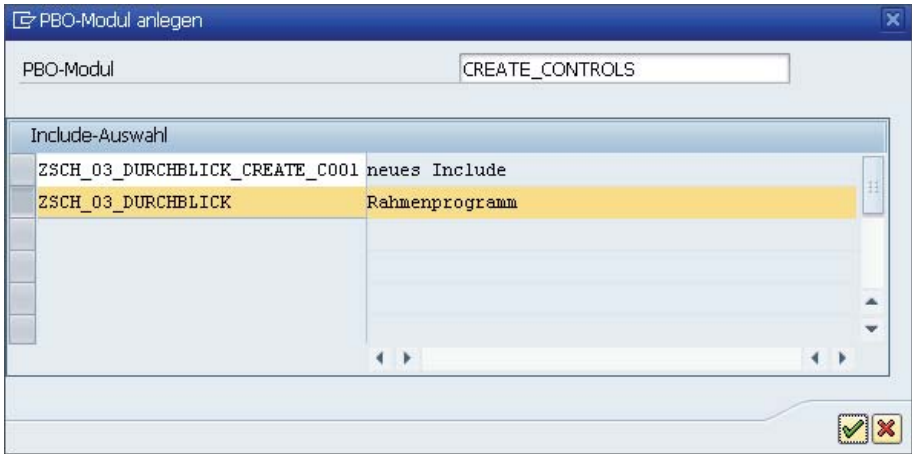


Sicher anlegen. Her damit.

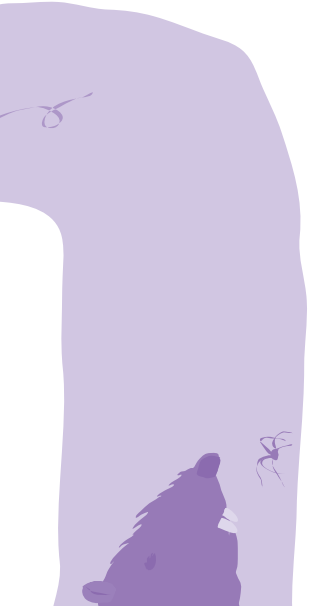
3. Der nächste Dialog fragt dich, wohin das Modul gestellt werden soll.

Entschuldigung, wir hätten da so ein Modul abzuliefern. 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 instanziieren
  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?

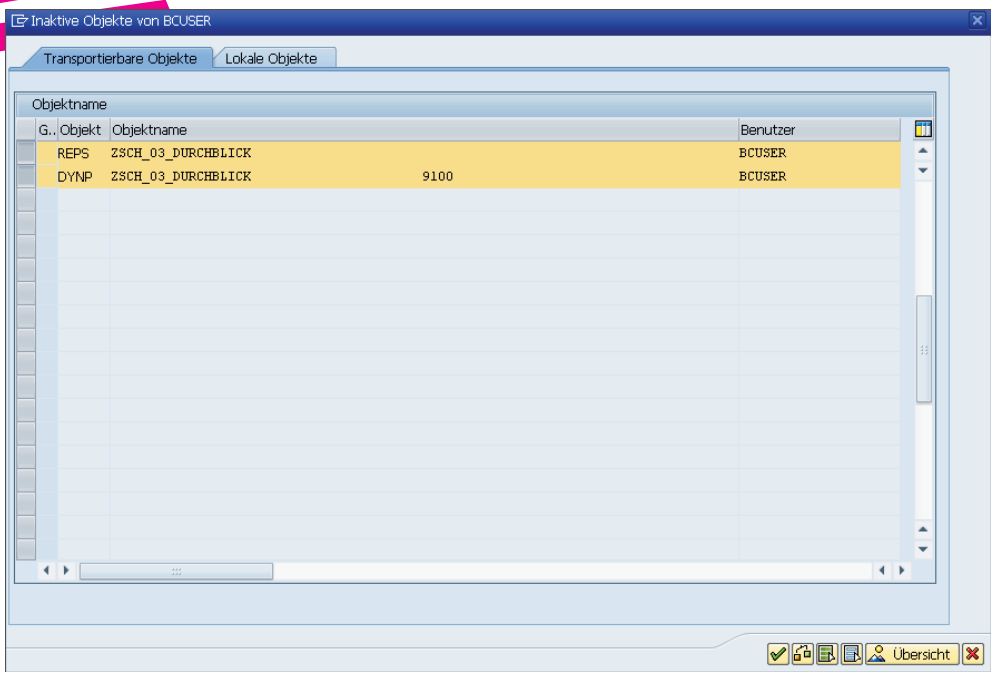
*2 Falls nicht, dann erzeuge ich ihn mit einer speziellen Funktion, `CREATE OBJECT`, und gebe den Namen `BILD` der **Custom Control Area** aus dem Layout mit. Damit weiß der Container, wofür er zuständig ist.

*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!

Tschinderassa-Tschniderassa-Tsching-Tsching-Tsching.

Trommelwirbel

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

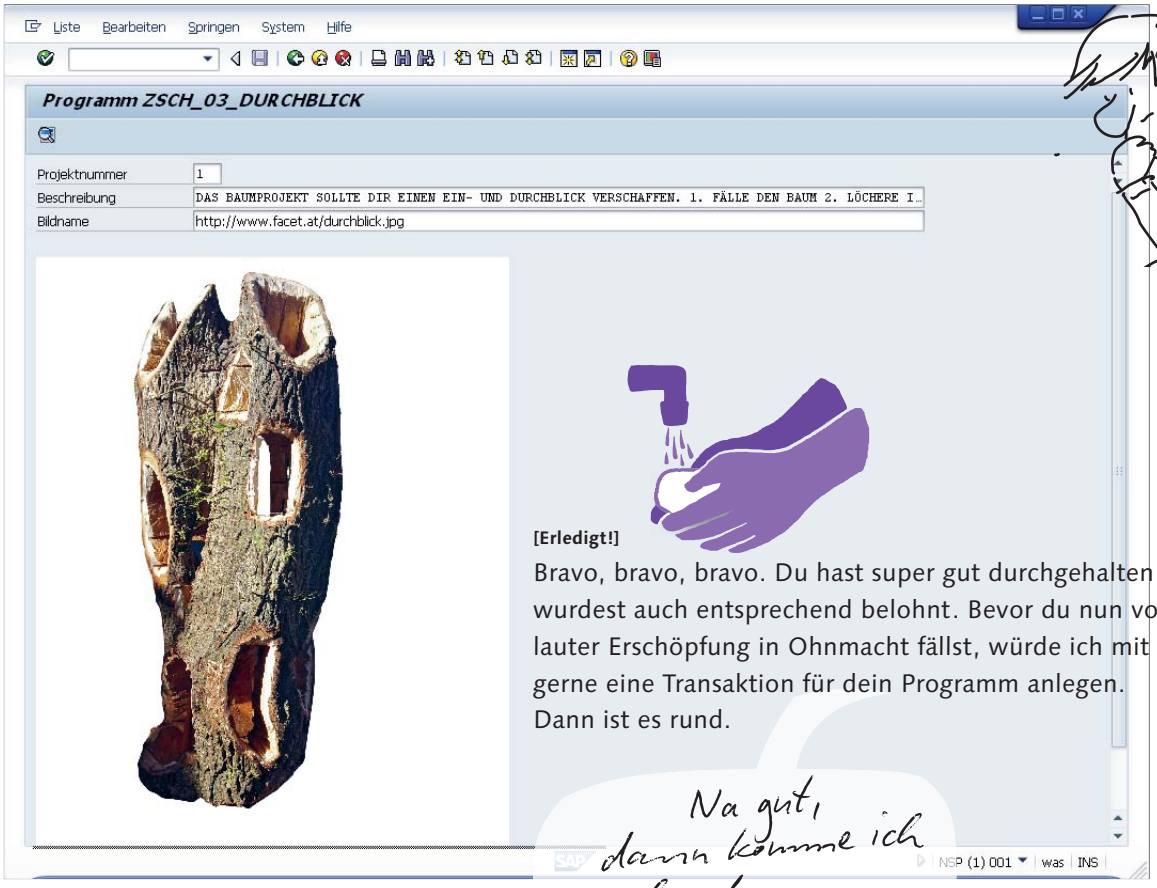
Das Selektionsbild, da gebe ich mal 001 ein.

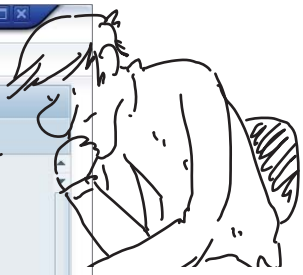
Hallo du – die Taste **Ausführen** (F8) – los – drücken!

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.





[Erledigt!]

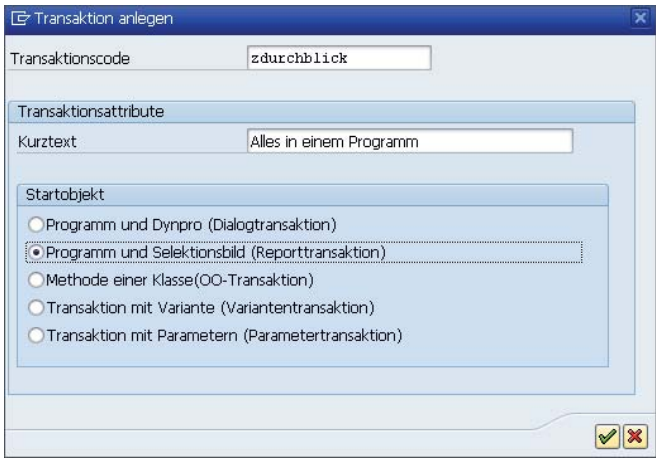
Bravo, bravo, bravo. Du hast super gut durchgehalten und wurdest auch entsprechend belohnt. Bevor du nun vor lauter Erschöpfung in Ohnmacht fällst, würde ich mit dir noch gerne eine Transaktion für dein Programm anlegen. Dann ist es rund.

Na gut, dann komme ich nochmal aus dem Sarg.

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

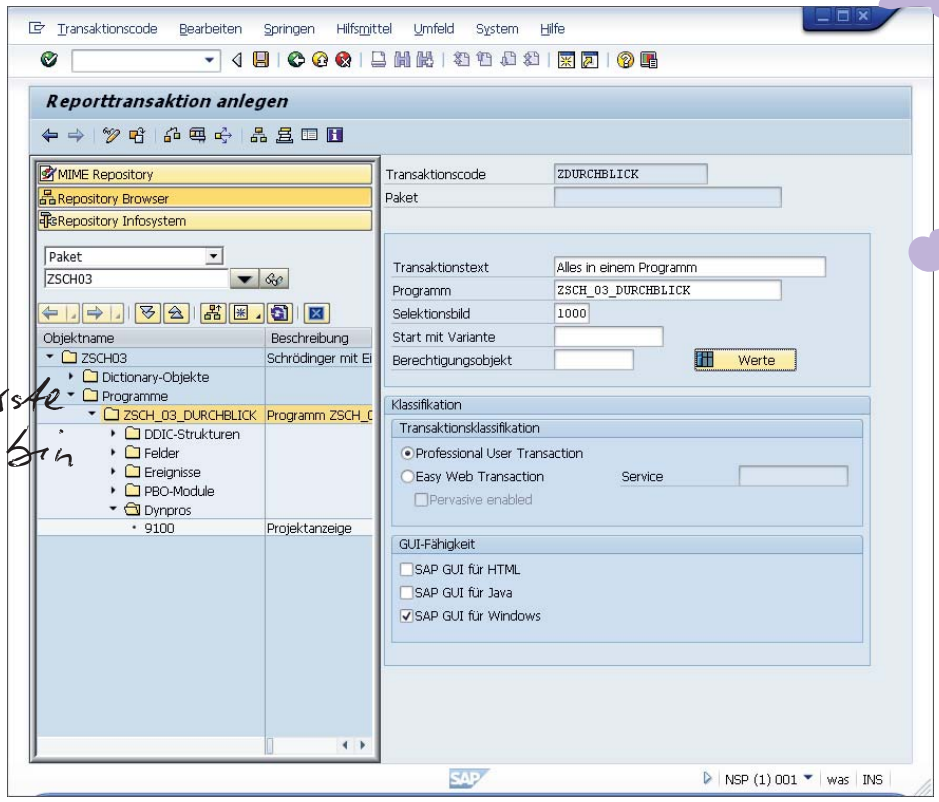
Die **Transaktion** (das ist eine ziemlich verwirrende Bezeichnung für einen Shortcut) ist der wohlverdiente Abschluss der Entwicklung. Damit denken wir, so wie natürlich immer, an den **Komfort des Anwenders**, da dieser die Transaktion aufrufen wird, um in den Genuss der von uns entwickelten Funktionalität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



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.

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 ;-)

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



Yippieh, 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

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z