

"Kannst du bitte eine Änderung in Zeile 7.656 machen?"

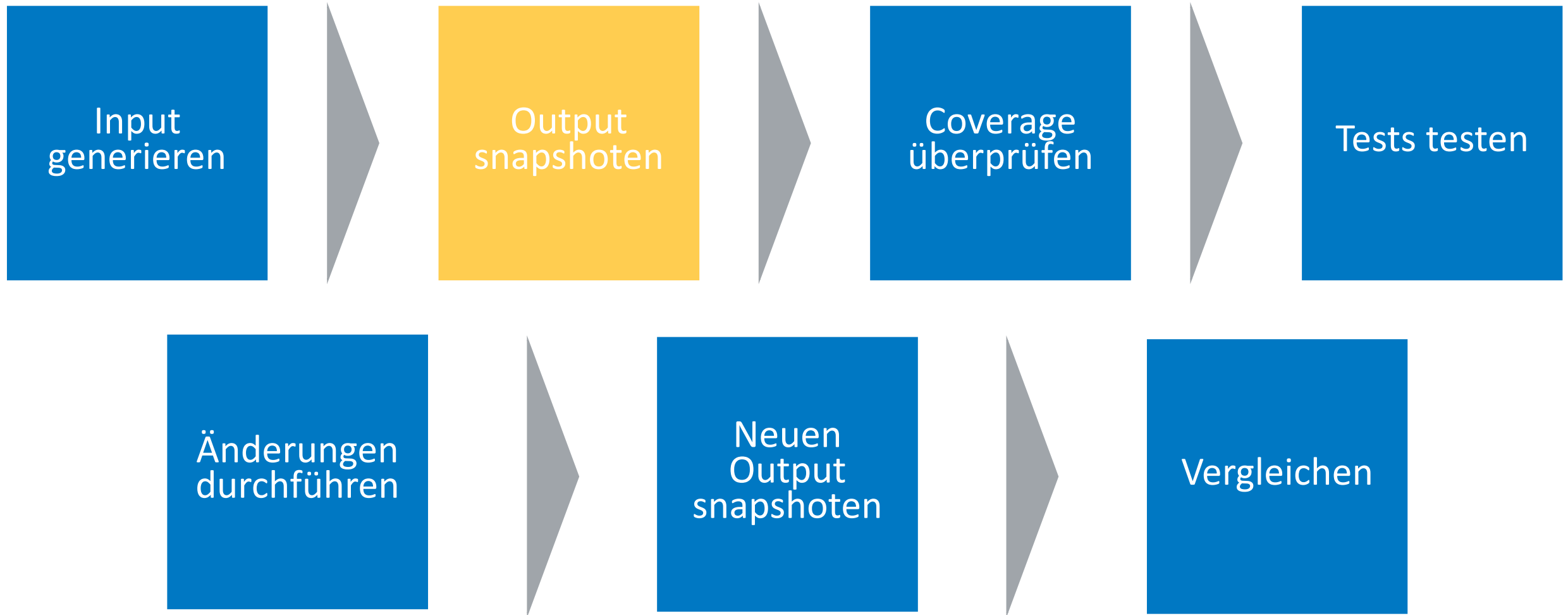
Legacy Code angstfrei mit der Golden Master Technik ändern

Dominik Panzer, INTENSE AG



PanzerDominik



Golden Master Testing im Überblick




Der einfachste Fall



Kundenliste mit Umsatz und Status






 

Business Partner bis 

☒ Protokollausgabe
☐ ALV Ausgabe
☒ Testmodus



Anzeigelimit

ABAP: Variantenkatalog des Programms ZSHOW_CUSTOMERS...

Variantenkatalog des Programms ZSHOW_CUSTOMER_STATUS

Variantenname	Kurzbeschreibung
PRODUKTION	Produktivvariante

Der einfachste Fall

Coverage Analyse mit SCOV



Coverage Analyzer: Ein/Aus, Status

- ▼ Coverage Analyzer
 - ▼ Administration
 - Ein/Aus, Status
 - Testgruppen
 - Registrierung
 - Zurücksetzen
 - Einstellungen
 - Monitor
 - Konsistenzprüfungen
 - > Anzeige

Ein-/Ausschalten des Coverage Analyzers

Status



Ein

Aus

 Detail


Ein-/Ausschalten der automatischen Aufzeichnung der Historie

Status



Ein

Aus

 Detail


Ein-/Ausschalten der Datenverdichtung aus verschiedenen Remote-Systemen

Status





Ein

Aus

 Detail



Status der Datensammlung auf den verschiedenen Servern

	Exception	AS-Instanz	Läuft	Batch
		SAPS4_S43_00	X	X

Golden Master mit kompaktem Output



Coverage Analyzer: Programmobjekte

▼ Coverage Analyzer

- Administration
 - Ein/Aus, Status
 - Testgruppen
 - Registrierung
 - Zurücksetzen
 - Einstellungen
 - Monitor
 - Konsistenzprüfungen
- Anzeige
 - Detail
 - Global

Programmobjekte

Status	VBlock-Abd	Zweig-Abd.	Anw-Abdeck	Objektname	Obj	Σ V.Blöcke	Σ Zweige	Σ Anweisung.	Programmän	Verantwortl.	
■	100,0	54,5	78,6	ZSHOW_CUSTOMER_STATUS	PROG	1	11	14	0	DPANZER	
						■	1	■	11	■	14

```
30 * ... and some more business logic might be here
31
32 IF writeout = abap_true.
33   LOOP AT businesspartnerswithstatus ASSIGNING <businesspartner>.
34     WRITE: / <businesspartner>-partner, <businesspartner>-name_last, <businesspartner>-name_first, <businesspartner>-revenue.
35   ENDLOOP.
36 ELSEIF alvout = abap_true.
37   cl_salv_table=>factory( IMPORTING r_salv_table = alv_table CHANGING t_table = businesspartnerswithstatus ).
38   alv_table->display( ).
39 ENDIF.
```

Golden Master mit kompaktem Output



Kundenliste mit Umsatz und Status

Kundenliste mit Umsatz und Status

2	Klotz	Jonathan	0
3	Klotz	Jonathan	0
10000020	Koothrappali	Rajesh	0
10000030			0
10000031	Holzschuh	Thomas	0
10000032			
1010000020	Schön		
1010000021	Wasser		
1010000022			
1010000023			

Kundenliste mit Umsatz und Status

Kundenliste mit Umsatz und Status

2	Klotz	Jonathan	0
3	Klotz	Jonathan	0
10000020	Koothrappali	Rajesh	0
10000030			0
10000031	Holzschuh	Thomas	0
10000032			0
1010000020	Schön	Gerhard	0
1010000021	Wasser	Anja	0
1010000022			0

Golden Master mit kompaktem Output



C:\Users\dPanzer\OneDrive - INTENSE AG\Desktop\Golden Master\Screenshot Golden Master.png



C:\Users\dPanzer\OneDrive - INTENSE AG\Desktop\Golden Master\Screenshot Change 001.png

Kundenliste mit Umsatz und Status				
2	Klotz	Jonathan		0
3	Klotz	Jonathan		0
10000020	Koothrappali	Rajesh		0
10000030				0
10000031	Holzschuh	Thomas		0
10000032				0
1010000020	Schön	Gerhard		0
1010000021	Wasser	Anja		0
1010000022				0
1010000023				0

Kundenliste mit Umsatz und Status				
2	Klotz	Jonathan		0
3	Klotz	Jonathan		0
10000020	Koothrappali	Rajesh		0
10000030				0
10000031	Holzschuh	Thomas		0
10000032				0
1010000020	Schön	Gerhard		0
1010000021	Wasser	Anja		0
1010000022				0

Golden Master mit großen Logs

Kundenliste mit Umsatz und Status		
Kundenliste mit Umsatz und Status		
1010000092	SRIEMER	0
1010000093	SRIEMER	0
1010000094	SRIEMER	0
1010000095	Müller	0
1010000096	SRIEMER	0
1010000097	SRIEMER	0
1010000098	SRIEMER	0
1010000099	SRIEMER	0
1010000100	SRIEMER	0
1010000101	SRIEMER	0
1010000102	Bonus	0
1010000103	Müller	0
1010000104	SRIEMER	0
1010000105	SRIEMER	0
1010000106	SRIEMER	0
1010000107	SRIEMER	0
1010000108	Sedewitz	0
1010000111	Wessel	0
1010000112	SRIEMER	0
1010000113	Hofstadter	0
1010000114	Bergner	0
1010000115	Petersen	0
1010000116	Sara	0
1010000117	Jäckel	0
1010000118	Wackel	0
1010000119	Wackler	0
1010000120	Meiersen	0
1010000121	von Arensdorf	0
1010000122	Zimmermann	0
1010000126	Winter	0
1010000173	Zimmermann	0
1010000188	von Arensdorf	0
1010000189	Wackler	0
1010000190	Sara	0
1010000212	Grunwald	0
1010000213	Kleinert	0
1010000214	Bamberger	0
1010000215	Kleinert	0
1010000216	Grunwald	0
1010000296	Bamberger	0
1010000303	Winter	0
1010000317	Spielmann	0
1010000375	Engler	0
1010000398	Mathe	0
1010000592	Test3	0
1010000594	Mathematik	0
1010000595	Herbst	0
1010000596	Köhler	0
1010000597	Winterkorn	0
1010000598	Watt	0
1010000599	Diel	0
1010000600	Kombi	0
	Karin die zweite	0

 Liste sichern in Datei... 

In welchem Format soll die Liste gesichert werden ?



☒ Unkonvertiert

☐ Text mit Tabulatoren

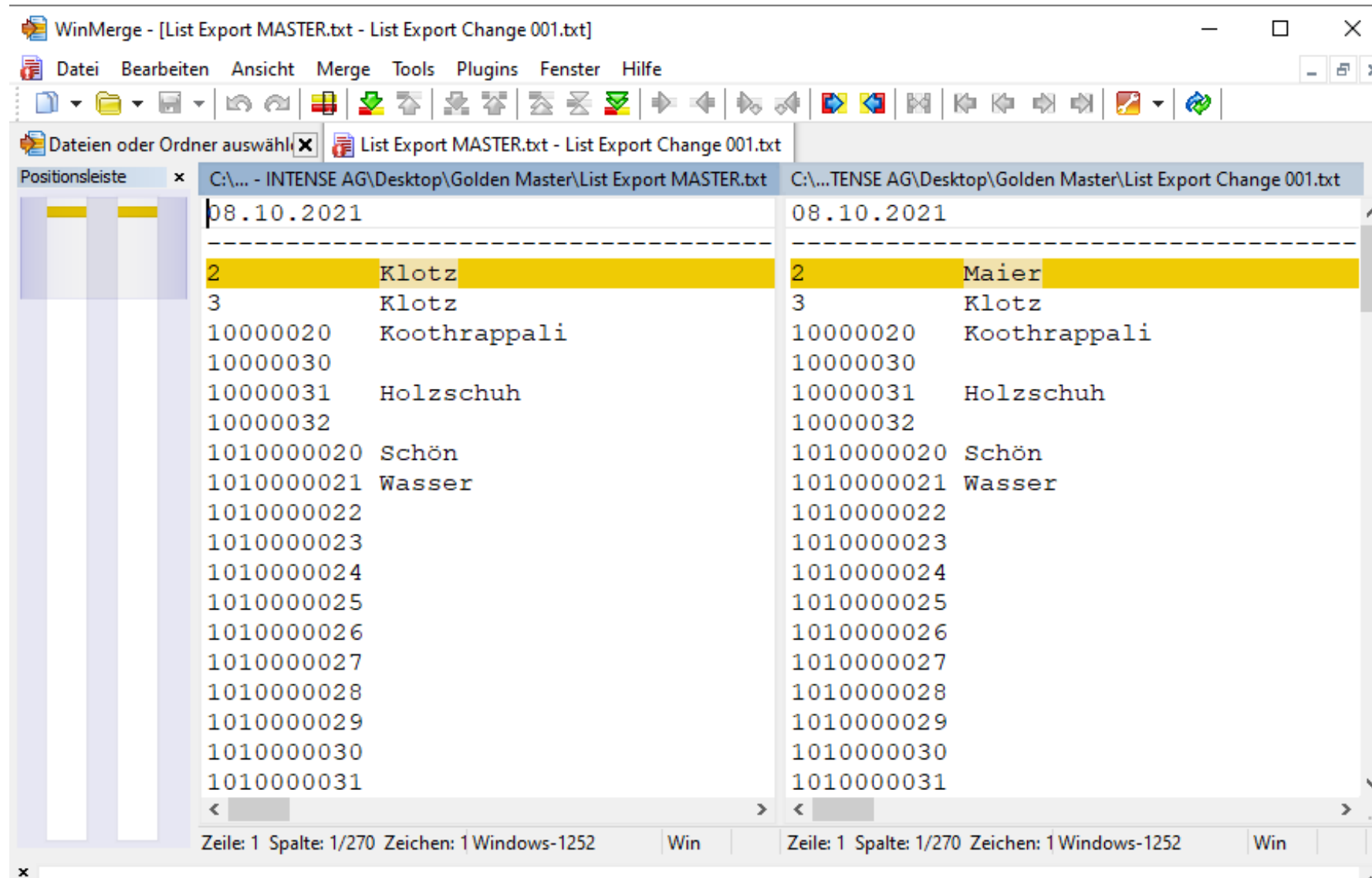
☐ Rich Text Format

☐ HTML Format

☐ In die Zwischenablage

Golden Master mit großen Logs



Beispiel 3: Golden Master mit ALV Output



GeschPartner	Nachname	Vorname	Revenue	is VIP
2	Klotz	Jonathan	0	
3	Klotz	Jonathan	0	
10000020	Koothrappali	Rajesh	0	
10000030			0	
10000031	Holzschuh	Thomas	0	
10000032			0	
1010000020	Schön	Gerhard	0	
1010000021	Wasser	Anja	0	
1010000022			0	
1010000023			0	
1010000024			0	
1010000025			0	
1010000026			12.000	X
1010000027			10.000	
1010000028			100	
1010000029			12.000	X
1010000030			10	
1010000031			50	
1010000032			9.942	
1010000033			0	
1010000034	Rupp	David	0	
1010000040	SRIEMER	SRIEMER	0	
1010000041	SRIEMER	SRIEMER	0	
1010000042	SRIEMER	SRIEMER	0	
1010000043	SRIEMER	SRIEMER	0	
1010000044	SRIEMER	SRIEMER	0	

GPartner	Nachname	Vorname	Revenue	is VIP
2	Klotz	Jonathan	0	
3	Klotz	Jonathan	0	
10000020	Koothrappali	Rajesh	0	
10000030			0	
10000031	Holzschuh	Thomas	0	
10000032			0	
1010000020	Schön	Gerhard	0	
1010000021	Wasser	Anja	0	
1010000022			0	
1010000023			0	
1010000024			0	
1010000025			0	
1010000026			12.000	X
1010000027			10.000	
1010000028			100	
1010000029			12.000	X
1010000030			10	
1010000031			50	
1010000032			9.942	
1010000033			0	
1010000034	Rupp	David	0	
1010000040	SRIEMER	SRIEMER	0	
1010000041	SRIEMER	SRIEMER	0	
1010000042	SRIEMER	SRIEMER	0	
1010000043	SRIEMER	SRIEMER	0	
1010000044	SRIEMER	SRIEMER	0	
1010000045	SRIEMER	SRIEMER	0	
1010000046	SRIEMER	SRIEMER	0	
1010000047	SRIEMER	SRIEMER	0	
1010000048	SRIEMER	SRIEMER	0	
1010000049	SRIEMER	SRIEMER	0	
1010000050			0	
1010000051			0	
1010000052			0	

Golden Master mit ALV Output



WinMerge - [20211008 Golde MASTER ALV 1.txt - 20211008 Golde MASTER ALV Change 1.txt]

Datei Bearbeiten Ansicht Merge Tools Plugins Fenster Hilfe

Dateien oder Ordner auswählen List Export MASTER.txt - List E... Dateien oder Ordner auswählen 20211008 Golde MASTER ALV 1.txt... Dateien oder Ordner auswählen

Positionenleiste x C:\...eDrive - INTENSE AG\Desktop\Golden Master\20211008 Golde MASTER ALV 1.txt C:\... INTENSE AG\Desktop\Golden Master\20211008 Golde MASTER ALV Change 1.txt

1010000214 Bamberger	Wolfhilde		1010000214 Bamberger	Wolfhilde	
1010000215 Kleinert	Geert		1010000215 Kleinert	Geert	
1010000216 Grunwald	Meta		1010000216 Grunwald	Meta	
1010000296 Bamberger	Wolfhilde		1010000296 Bamberger	Wolfhilde	
1010000303 Winter	Lars		1010000303 Winter	Lars	
1010000317 Spielmann	Beate		1010000317 Spielmann	Beate	
1010000375 Engler	Eckbert		1010000375 Engler	Eckbert	
1010000398 Mathe	Lisa		1010000398 Mathe	Lisa-Lena	
1010000592 Test3	Test		1010000592 Test3	Test	
1010000594 Mathematik	Lisa				
1010000595 Herbst	Franziska		1010000595 Herbst	Franziska	
1010000596 Köhler	Franziska		1010000596 Köhler	Franziska	
1010000597 Winterkorn	Franziska		1010000597 Winterkorn	Franziska	
1010000598 Watt	Kilo-Mega		1010000598 Watt	Kilo-Mega	
1010000599 Diel	Karola				
1010000600 Kombi	Karin die zweite				

Zeile: 115 Spalte: 1/59 Zeichen: 1/59 Windows-1252 Win Zeile: 115-116 Windows-1252 Win



Automatisierung mit ABAP Unit und klassischen Reports

Testing isoliertes Form mit Rückgabe

```
1 REPORT zunit_test_forms.
2
3 START-OF-SELECTION.
4
5   DATA: summe TYPE int4.
6
7   PERFORM addiere USING 1 2 CHANGING summe.
8
9 FORM addiere USING summand1 summand2 CHANGING summe.
10   summe = summand1 + summand2.
11 ENDFORM.
12
13
14
15 CLASS addierer_tests DEFINITION FINAL FOR TESTING
16   RISK LEVEL HARMLESS.
17   PRIVATE SECTION.
18     METHODS: akz_addiere_1_und_2_gleich_3 FOR TESTING RAISING cx_static_check.
19   ENDCLASS.
20
21 CLASS addierer_tests IMPLEMENTATION.
22   METHOD akz_addiere_1_und_2_gleich_3.
23     DATA: summe_actual TYPE int4.
24
25     PERFORM addiere IN PROGRAM zunit_test_forms USING 1 2 CHANGING summe_actual.
26
27     cl_abap_unit_assert=>assert_equals( exp = 3 act = summe_actual ).
28   ENDMETHOD.
29   ENDCLASS.
```

Automatisierung mit ABAP Unit und klassischen Reports

Testing lokale Klasse mit Rückgabe

```
3 CLASS addierer DEFINITION.  
4   PUBLIC SECTION.  
5     METHODS: addiere IMPORTING summand1      TYPE int4  
6                     summand2      TYPE int4  
7                     RETURNING VALUE(summe) TYPE int4.  
8 ENDCLASS.  
9  
10 CLASS addierer IMPLEMENTATION.  
11   METHOD addiere.  
12     summe = summand1 + summand2.  
13   ENDMETHOD.  
14 ENDCLASS.  
15  
16  
17 START-OF-SELECTION.  
18   DATA(addierer) = NEW addierer( ).  
19   DATA(summe) = addierer->addiere(  
20     summand1 = 1  
21     summand2 = 2 ).  
22   WRITE summe.  
23
```

```
24  
25 CLASS addierer_tests DEFINITION FINAL FOR TESTING  
26   RISK LEVEL HARMLESS.  
27   PRIVATE SECTION.  
28     METHODS: akz_addiere_1_und_2_gleich_3 FOR TESTING RAISING cx_static_check,  
29     setup.  
30     DATA: addierer TYPE REF TO addierer.  
31 ENDCLASS.  
32  
33  
34 CLASS addierer_tests IMPLEMENTATION.  
35   METHOD setup.  
36     addierer = NEW addierer( ).  
37   ENDMETHOD.  
38  
39   METHOD akz_addiere_1_und_2_gleich_3.  
40  
41     DATA(summe_actual) = addierer->addiere(  
42       summand1 = 1  
43       summand2 = 2 ).  
44     cl_abap_unit_assert=>assert_equals( exp = 3 act = summe_actual ).  
45   ENDMETHOD.  
46 ENDCLASS.
```



Automatisierung mit ABAP Unit und klassischen Reports

Testing ohne Strukturierung mit CALL TRANSACTION

```
1 REPORT zunit_test_no_structure.  
2  
3 PARAMETERS: sum1 TYPE int4,  
4             sum2 TYPE int4.  
5  
6 START-OF-SELECTION.  
7   DATA(summe) = sum1 + sum2.  
8   IF summe = 0.  
9     MESSAGE |Fehler Summe ist 0| TYPE 'E'.  
10  ELSE.  
11    WRITE summe.  
12  ENDIF.  
13
```



Automatisierung mit ABAP Unit und klassischen Reports

Testing ohne Strukturierung mit CALL TRANSACTION

```
15
16 CLASS addierer_tests DEFINITION FINAL FOR TESTING
17   RISK LEVEL HARMLESS.
18   PRIVATE SECTION.
19     METHODS: addiere_call_transaction FOR TESTING.
20 ENDCLASS.
21
22 CLASS addierer_tests IMPLEMENTATION.
23   METHOD addiere_call_transaction.
24     DATA: messages TYPE STANDARD TABLE OF bdcmsgcoll.
25
26     DATA(batchdata) = VALUE bdcdata_tab( ( program = 'ZUNIT_TEST_NO_STRUCTURE' dynpro = '1000' dynbegin = 'X' fnam = '' fval = '' )
27                                           ( program = 'ZUNIT_TEST_NO_STRUCTURE' dynpro = '1000' dynbegin = '' fnam = 'SUM1' fval = '0' )
28                                           ( program = 'ZUNIT_TEST_NO_STRUCTURE' dynpro = '1000' dynbegin = '' fnam = 'SUM2' fval = '0' )
29                                           ( program = '' dynpro = '' dynbegin = '' fnam = 'BDC_OKCODE' fval = '=ONLI' ) ).
30
31     CALL TRANSACTION 'ZUNIT_NO_STRUCTURE'
32       USING batchdata
33       MODE 'N' UPDATE 'A'|
34       MESSAGES INTO messages.
35
36     cl_aunit_assert=>assert_equals(
37       EXPORTING
38         exp          = |Fehler Summe ist 0|
39         act          = messages[ 1 ]-msgv1 ).
40   ENDMETHOD.
41 ENDCLASS.
```



ABER was ist denn wenn...

- Der Golden Master groß ist?
- Es keinen Output gibt, den man einfach abgreifen könnte?
- Zufallszahlen involviert sind?
- GUIDs und Nummernkreisobjekt verwendet werden?
- Abfragen auf SY-DATUM stattfinden?
- Externe APIs angesprochen werden?
- Datenbankänderungen vorgenommen werden?
- ...



„Nahtstellen“ identifizieren, an denen „relativ sicher“
Änderungen vorgenommen werden können

Wichtig: Änderungen mitprotokollieren



Golden Master Daten importieren

- Hart codiert im Test
- In Unit Tests über Import aus File / Cluster Tabelle

```
51 DATA(golden_master) = zread_golden_master=>read_master_table( key = 'GM1' ).
52
53 * zu testendes Coding aufrufen
54
55 cl_abap_unit_assert=>assert_equals(
56   EXPORTING
57     act      = actual_result
58     exp      = golden_master ).
```



Output erzeugen

- Write Statements einfügen
- ALV einfügen
- Messages ausgeben
- Return-Parameter einfügen
- Memory-Export / Import
- Puffer-Datenbank nutzen
- Cluster-Tabellen nutzen
- Eigenen Logger aufrufen
- ...



Wie mit Datenbankänderungen umgehen?

- INSERTs / MODIFY etc. können problemlos sein
- Auskommentieren
- Durch eigene Datenbanktabellen ersetzen
- explizites Löschen bei INSERTs



GUIDS / Nummernkreisobjekte / Zufallszahlen / APIs

1. GUIDS / Nummernkreise nicht in der Vergleich mit dem Golden Master mit einbeziehen
2. Nummernkreisobjekte nach jedem Lauf resettet
3. GUIDs statisch zurückgeben
4. Zufallszahlen durch Festwerte ersetzen
5. SY-DATUM durch Festwert ersetzen
6. APIS deaktivieren / durch Festwerte ersetzen
7. Außer: API Calls sind unser Ergebnis! Dann protokollieren.



Abhängigkeiten durch Fake-Daten ersetzen

Über Interfaces, ohne Framework

```
9  START-OF-SELECTION.  
10  
11  DATA: systemdatum TYPE dats.  
12  
13  systemdatum = sy-datum.  
14  WRITE / systemdatum.
```

```
1● INTERFACE zif_kalender  
2  PUBLIC .  
3  METHODS: get_heute RETURNING VALUE(heute) TYPE dats.  
4  
5  ENDINTERFACE.
```



Abhängigkeiten durch Fake-Daten ersetzen

Über Interfaces, ohne Framework

```
1● CLASS zkalender DEFINITION
2    PUBLIC
3    FINAL
4    CREATE PUBLIC.
5
6    PUBLIC SECTION.
7        INTERFACES zif_kalender.
8    PROTECTED SECTION.
9    PRIVATE SECTION.
10 ENDCLASS.
11
12
13
14● CLASS zkalender IMPLEMENTATION.
15● METHOD zif_kalender~get_heute.
16     heute = sy-datum.
17 ENDMETHOD.
18
19 ENDCLASS.
```








```
1● CLASS zkalender_fake_christmas DEFINITION
2    PUBLIC
3    FINAL
4    CREATE PUBLIC.
5
6    PUBLIC SECTION.
7        INTERFACES zif_kalender.
8    PROTECTED SECTION.
9    PRIVATE SECTION.
10 ENDCLASS.
11
12
13
14● CLASS zkalender_fake_christmas IMPLEMENTATION.
15● METHOD zif_kalender~get_heute.
16     CONSTANTS: christmas TYPE dats VALUE '20211224'.
17     heute = christmas.
18 ENDMETHOD.
19
20 ENDCLASS.
```



Abhängigkeiten durch Fake-Daten ersetzen

Über Interfaces, ohne Framework

```
9  START-OF-SELECTION.  
10  
11  DATA: systemdatum TYPE dats.  
12  
13  systemdatum = sy-datum.  
14  WRITE / systemdatum.  
15  
16  
17  DATA: kalender TYPE REF TO zif_kalender.  
18  
19  kalender = NEW zkalender( ).  
20  systemdatum = kalender->get_heute( ).  
21  WRITE / systemdatum.  
22  
23  
24  kalender = NEW zkalender_fake_christmas( ).  
25  systemdatum = kalender->get_heute( ).  
26  WRITE / systemdatum.
```

Menü       

Systemdatum faken

Systemdatum faken

26.10.2021
26.10.2021
24.12.2021



Abhängigkeiten durch Fake-Daten ersetzen

Mit Test Double Framework

```
9  START-OF-SELECTION.  
10  
11  DATA: systemdatum TYPE dats.  
12  
13  systemdatum = sy-datum.  
14  WRITE / systemdatum.
```

```
1● INTERFACE zif_kalender  
2  PUBLIC .  
3  METHODS: get_heute RETURNING VALUE(heute) TYPE dats.  
4  
5  ENDINTERFACE.
```




Abhängigkeiten durch Fake-Daten ersetzen

Mit Test Double Framework

```
39 METHOD tagesdatum_ist_Jahresanfang.  
40   DATA: kalender_double TYPE REF TO zif_kalender.  
41   CONSTANTS: Jahresanfang TYPE dats VALUE '20210101'.  
42  
43   kalender_double ?= cl_abap_testdouble=>create( 'zif_kalender').  
44   cl_abap_testdouble=>configure_call( kalender_double )->returning( Jahresanfang ).  
45   kalender_double->get_heute( ).  
46  
47   DATA(tagesdatum) = kalender_double->get_heute( ).  
48  
49   cl_abap_unit_assert=>assert_equals( exp = Jahresanfang act = tagesdatum ).  
50 ENDMETHOD.  
51 ENDCLASS.
```

```
26 METHOD tagesdatum_ist_geburtstag.  
27   DATA: kalender_double TYPE REF TO zif_kalender.  
28   CONSTANTS: geburtstag TYPE dats VALUE '19821007'.  
29  
30   kalender_double ?= cl_abap_testdouble=>create( 'zkalender').  
31   cl_abap_testdouble=>configure_call( kalender_double )->returning( geburtstag ).  
32   kalender_double->get_heute( ).  
33  
34   DATA(tagesdatum) = kalender_double->get_heute( ).  
35  
36   cl_abap_unit_assert=>assert_equals( exp = geburtstag act = tagesdatum ).  
37 ENDMETHOD.
```



Abhängigkeiten durch Fake-Daten ersetzen

Mit Test Seams

```
3 START-OF-SELECTION.  
4  
5 DATA: summe TYPE int4.  
6  
7 PERFORM addiere USING 1 2 CHANGING summe.  
8 WRITE / summe.  
9 WRITE / sy-datum.  
10  
11 FORM addiere USING summand1 summand2 CHANGING summe.  
12 summe = summand1 + summand2.  
13 ENDFORM.
```

```
3 START-OF-SELECTION.  
4  
5 DATA: summe TYPE int4.  
6  
7 PERFORM addiere USING 1 2 CHANGING summe.  
8 WRITE / summe.  
9 ⚠ TEST-SEAM tagesdatum.  
10 WRITE / sy-datum.  
11 end-test-seam.  
12  
13 FORM addiere USING summand1 summand2 CHANGING summe.  
14 summe = summand1 + summand2.  
15 ENDFORM.
```



Abhängigkeiten durch Fake-Daten ersetzen

Mit Test Seams

```
1● CLASS zkalender_test_seam DEFINITION
2   PUBLIC
3   FINAL
4   CREATE PUBLIC.
5
6   PUBLIC SECTION.
7     INTERFACES zif_kalender.
8   PROTECTED SECTION.
9   PRIVATE SECTION.
10  ENDCLASS.
11
12
13
14● CLASS zkalender_test_seam IMPLEMENTATION.
15●   METHOD zif_kalender~get_heute.
16●     TEST-SEAM tagesdatum.
17     heute = sy-datum.
18     end-test-seam.
19   ENDMETHOD.
20
21  ENDCLASS.
```

```
1● CLASS kalendertest DEFINITION FINAL FOR TESTING
2   DURATION SHORT
3   RISK LEVEL HARMLESS.
4
5   PRIVATE SECTION.
6     METHODS:
7       tagesdatum_ist_weihnachten FOR TESTING RAISING cx_static_check.
8  ENDCLASS.
9
10
11● CLASS kalendertest IMPLEMENTATION.
12
13●   METHOD tagesdatum_ist_weihnachten.
14     CONSTANTS: christmas2021 TYPE dats VALUE '20211224'.
15
16●   TEST-INJECTION tagesdatum.
17     heute = '20211224'.
18   end-test-injection.
19
20     DATA(tagesdatum) = NEW zkalender_test_seam( )->zif_kalender~get_heute( ).
21
22     cl_abap_unit_assert=>assert_equals( exp = christmas2021 act = tagesdatum ).
23   ENDMETHOD.
24
25  ENDCLASS.
```



Code ist jetzt unter Test – wie geht's weiter?

Bloß nicht wie früher!

„Verlasse den Code sauberer als du ihn
vorgefunden hast.“



Code ist jetzt unter Test – wie geht's weiter?

Code verbessern ohne funktionale Änderung

1. Erkenntnisse festhalten

- Forms / Methoden bedeutsam umbenennen
- Variablen sprechend benennen
- Kommentare einfügen
- Doku schreiben

2. Ordnung schaffen

- Methoden extrahieren
- Auskommentiertes Coding entfernen
- Formatierung korrigieren
- Lesbarkeit erhöhen (EQ, GT etc.)

3. Testbarkeit verbessern

- Datenbankzugriffe in Klasse extrahieren
- API Calls in Klasse extrahieren



Code ist jetzt unter Test – wie geht's weiter?

Neue Funktionen einfügen – Sprout Method / Sprout Class

```
19 CLASS zsome_demo_class IMPLEMENTATION.  
20 METHOD do_something.  
21  
22 * ... complex code here  
23  
24 LOOP AT list ASSIGNING FIELD-SYMBOL(<listitem>).  
25     process( <listitem> ).  
26     save( <listitem> ).  
27 ENDLOOP.  
28  
29 * ...some more code here  
30 ENDMETHOD.
```

```
13 METHOD deduplicate_empty_list.  
14     DATA: list          TYPE tt_evopd,  
15           expectedlist TYPE tt_evopd.  
16  
17     DATA(democlass) = NEW zsome_demo_class( ).  
18  
19     DATA(deduplicatedlist) = democlass->deduplicate( list ).  
20  
21     cl_abap_unit_assert=>assert_equals( exp = expectedlist act = deduplicatedlist ).  
22 ENDMETHOD.
```



Code ist jetzt unter Test – wie geht's weiter?

Neue Funktionen einfügen – Sprout Method / Sprout Class

```
25 METHOD do_something.  
26  
27 * ... complex code here  
28  
29 DATA(deduplicatedlist) = deduplicate( list ).  
30  
31 LOOP AT deduplicatedlist ASSIGNING FIELD-SYMBOL(<listitem>).  
32     process( <listitem> ).  
33     save( <listitem> ).  
34 ENDLOOP.  
35  
36 * ...some more code here  
37 ENDMETHOD.
```

```
24 CLASS zsome_demo_class IMPLEMENTATION.  
25 METHOD do_something.  
26  
27 * ... complex code here  
28  
29 DATA(deduplicatedlist) = zdeduplicator=>deduplicatedlist( list ).  
30  
31 LOOP AT deduplicatedlist ASSIGNING FIELD-SYMBOL(<listitem>).  
32     process( <listitem> ).  
33     save( <listitem> ).  
34 ENDLOOP.  
35  
36 * ...some more code here  
37 ENDMETHOD.
```



Infos und Kontakt

Mehr Infos:

- [Legacy of Socrates Konferenz](#)
- [Michael C. Feathers - Working Effectively with Legacy Code](#)
- https://en.wikipedia.org/wiki/Characterization_test

Tools:

- [WinMerge](#)

Fragen?

Verbesserungsvorschläge?

Philosophieren?

Lust auf einen praktischen Workshop?

Usergroup Vortrag?

Gerne!

Dominik.Panzer@intense.de



PanzerDominik