FORMS

Wszystkie komendy forms

Form Builder maintains the values of system variables on a per form basis. That is, the values of all system variables correspond only to the current form. The names of the available system variables are:

- SYSTEM.BLOCK_STATUS
- SYSTEM.COORDINATION_OPERATION
- SYSTEM.CURRENT_BLOCK
- SYSTEM.CURRENT_DATETIME

- SYSTEM.CURRENT_FORM
- SYSTEM.CURRENT_ITEM
- SYSTEM.CURRENT_VALUE
- SYSTEM.CURSOR_BLOCK
- SYSTEM.CURSOR_ITEM
- SYSTEM.CURSOR_RECORD
- SYSTEM.CURSOR_VALUE
- SYSTEM.CUSTOM_ITEM_EVENT
- SYSTEM.CUSTOM_ITEM_EVENT_PARAMETERS
- SYSTEM.DATE_THRESHOLD*
- SYSTEM.EFFECTIVE_DATE*
- SYSTEM.EVENT_WINDOW
- SYSTEM.FORM_STATUS
- SYSTEM.LAST_QUERY
- SYSTEM.LAST_RECORD
- SYSTEM.MASTER_BLOCK
- SYSTEM.MESSAGE_LEVEL*
- SYSTEM.MODE
- SYSTEM.MOUSE_BUTTON_PRESSED
- SYSTEM.MOUSE_BUTTON_SHIFT_STATE
- SYSTEM.MOUSE_ITEM
- SYSTEM.MOUSE_CANVAS
- SYSTEM.MOUSE_X_POS
- SYSTEM.MOUSE_Y_POS
- SYSTEM.MOUSE_RECORD
- SYSTEM.MOUSE_RECORD_OFFSET
- SYSTEM.RECORD_STATUS
- SYSTEM.SUPPRESS_WORKING*
- SYSTEM.TAB_NEW_PAGE
- SYSTEM.TAB_PREVIOUS_PAGE

- SYSTEM.TRIGGER_BLOCK
- SYSTEM.TRIGGER_ITEM
- SYSTEM.TRIGGER_RECORD

A
ABORT_QUERY
ADD_GROUP_COLUMN
ADD_GROUP_ROW
ADD_LIST_ELEMENT
ADD_OLEARGS
ADD_PARAMETER
APPLICATION_PARAMETER

B
BELL
BLOCK_MENU
BREAK

C
CALL_FORM
CALL_INPUT
CALL_OLE
CALL_OLE_<return type>
CANCEL_REPORT_OBJECT
CHECK_RECORD_UNIQUENESS

CHECKBOX_CHECKED
CHECKED
CLEAR_BLOCK
CLEAR_EOL
CLEAR_FORM
CLEAR_ITEM
CLEAR_LIST
CLEAR_MESSAGE
CLEAR_RECORD
CLOSE_FORM
COMMIT_FORM
CONVERT_OTHER_VALUE
COPY
COPY_REGION
COPY_REPORT_OUTPUT
COUNT_QUERY
CREATE_GROUP
CREATE_GROUP_FROM_QUERY
CREATE_OLEOBJ

CREATE_PARAMETER_LIST
CREATE_QUERIED_RECORD
CREATE_RECORD
CREATE_TIMER
CREATE_VAR
CUT_REGION

D
DBMS_ERROR_CODE
DBMS_ERROR_TEXT

DEBUG_MODE
DEFAULT_VALUE
DELETE_GROUP
DELETE_GROUP_ROW
DELETE_LIST_ELEMENT
DELETE_PARAMETER
DELETE_RECORD
DELETE_TIMER
DESTROY_PARAMETER_LIST

DESTROY_VARIANT
DISPATCH_EVENT
DISPLAY_ERROR
DISPLAY_ITEM
DOWN
DO_KEY
DUMMY_REFERENCE
DUPLICATE_ITEM
DUPLICATE_RECORD

E
EDIT_TEXTITEM
ENFORCE_COLUMN_SECURITY
ENTER
ENTER_QUERY
ERASE
ERROR_CODE
ERROR_TEXT
ERROR_TYPE
EXECUTE_QUERY

EXECUTE_TRIGGER
EXIT_FORM

F
FETCH_RECORDS
FIND_ALERT
FIND_BLOCK
FIND_CANVAS
FIND_COLUMN
FIND_EDITOR
FIND_FORM
FIND_GROUP
FIND_ITEM
FIND_LOV
FIND_MENU_ITEM
FIND_RELATION
FIND_REPORT_OBJECT
FIND_TAB_PAGE
FIND_TIMER
FIND_VIEW

FIND_WINDOW
FIRST_RECORD
FORM_FAILURE
FORM_FATAL
FORM_SUCCESS
FORMS_DDL
FORMS_OLE.ACTIVATE_SERVER
FORMS_OLE.CLOSE_SERVER

FORMS_OLE.EXEC_VERB
FORMS_OLE.FIND_OLE_VERB
FORMS_OLE.GET_INTERFACE_POINTER
FORMS_OLE.GET_VERB_COUNT
FORMS_OLE.GET_VERB_NAME
FORMS_OLE.INITIALIZE_CONTAINER
FORMS_OLE.SERVER_ACTIVE

G
GENERATE_SEQUENCE_NUMBER
GET_APPLICATION_PROPERTY
GET_BLOCK_PROPERTY
GET_CANVAS_PROPERTY
GET_FILE_NAME
GET_FORM_PROPERTY
GET_GROUP_CHAR_CELL
GET_GROUP_DATE_CELL
GET_GROUP_NUMBER_CELL
GET_GROUP_RECORD_NUMBER
GET_GROUP_ROW_COUNT
GET_GROUP_SELECTION
GET_GROUP_SELECTION_COUNT
GET_INTERFACE_POINTER

GET_ITEM_INSTANCE_PROPERTY
GET_ITEM_PROPERTY
GET_LIST_ELEMENT_COUNT
GET_LIST_ELEMENT_LABEL
GET_LIST_ELEMENT_VALUE
GET_LOV_PROPERTY
GET_MENU_ITEM_PROPERTY
GET_MESSAGE
GET_OLEARG_<type>
GET_OLE_MEMBERID
GET_OLE_<proptype>
GET_PARAMETER_ATTR
GET_PARAMETER_LIST
GET_RADIO_BUTTON_PROPERTY
GET_RECORD_PROPERTY

GET_RELATION_PROPERTY
GET_REPORT_OBJECT_PROPERTY
GET_TAB_PAGE_PROPERTY
GET_VAR_BOUNDS
GET_VAR_DIMS
GET_VAR_TYPE
GET_VIEW_PROPERTY
GET_WINDOW_PROPERTY
GO_BLOCK
GO_FORM
GO_ITEM
GO_RECORD

H
HELP
HIDE_MENU
HIDE_VIEW
HIDE_WINDOW
HOST

I

ID_NULL
IMAGE_SCROLL
IMAGE_ZOOM
INIT_OLEARGS
INITIALIZE_CONTAINER
INSERT_RECORD
ISSUE_ROLLBACK
ISSUE_SAVEPOINT
ITEM_ENABLED

J

K

L
LAST_OLE_ERROR
LAST_OLE_EXCEPTION
LAST_RECORD
LIST_VALUES
LOCK_RECORD
LOGON
LOGON_SCREEN

LOGOUT

M
MENU_CLEAR_FIELD
MENU_NEXT_FIELD
MENU_PARAMETER
MENU_PREVIOUS_FIELD
MENU_REDISPLAY
MENU_SHOW_KEYS
MESSAGE
MESSAGE_CODE
MESSAGE_TEXT
MESSAGE_TYPE
MOVE_WINDOW

N
NAME_IN
NEW_FORM
NEXT_BLOCK
NEXT_ITEM
NEXT_FORM
NEXT_KEY

NEXT_MENU_ITEM
NEXT_RECORD
NEXT_SET

O
OLEVAR_EMPTY
OPEN_FORM

P
PASTE_REGION
PAUSE
PECS.ADD_CLASS

PECS.ADD_EVENT
PECS.COLLECTOR
PECS.DISABLE_CLASS
PECS.ENABLE_CLASS
PECS.END_EVENT
PECS.POINT_EVENT
PECS.START_EVENT

PLAY_SOUND
POPULATE_GROUP
POPULATE_GROUP_WITH_QUERY
POPULATE_LIST
POST
PREVIOUS_BLOCK
PREVIOUS_FORM
PREVIOUS_ITEM
PREVIOUS_MENU
PREVIOUS_MENU_ITEM
PREVIOUS_RECORD
PRINT
PTR_TO_VAR

Q
QUERY_PARAMETER

R
READ_IMAGE_FILE
READ_SOUND_FILE
RECALCULATE

REDISPLAY
RELEASE_OBJ
REPLACE_CONTENT_VIEW
REPLACE_MENU
REPORT_OBJECT_STATUS
RESET_GROUP_SELECTION
RESIZE_WINDOW
RETRIEVE_LIST
RUN_PRODUCT
RUN_REPORT_OBJECT

S
SCROLL_DOWN
SCROLL_UP
SCROLL_VIEW
SELECT_ALL
SELECT_RECORDS
SET_ALERT_BUTTON_PROPERTY

SET_ALERT_PROPERTY
SET_APPLICATION_PROPERTY
SET_BLOCK_PROPERTY
SET_CANVAS_PROPERTY
SET_FORM_PROPERTY
SET_GROUP_CHAR_CELL
SET_GROUP_DATE_CELL
SET_GROUP_NUMBER_CELL
SET_GROUP_SELECTION
SET_INPUT_FOCUS
SET_ITEM_INSTANCE_PROPERTY

SET_ITEM_PROPERTY
SET_LOV_COLUMN_PROPERTY
SET_LOV_PROPERTY
SET_MENU_ITEM_PROPERTY

SET_OLE
SET_PARAMETER_ATTR
SET_RADIO_BUTTON_PROPERTY
SET_RECORD_PROPERTY
SET_RELATION_PROPERTY
SET_REPORT_OBJECT_PROPERTY
SET_TAB_PAGE_PROPERTY
SET_TIMER
SET_VAR
SET_VIEW_PROPERTY
SET_WINDOW_PROPERTY
SHOW_ALERT
SHOW_EDITOR
SHOW_KEYS
SHOW_LOV
SHOW_MENU
SHOW_VIEW

SHOW_WINDOW
SYNCHRONIZE

T
TERMINATE
TO_VARIANT

U
UNSET_GROUP_SELECTION
UP
UPDATE_CHART
UPDATE_RECORD
USER_EXIT

V
VALIDATE
VARPTR_TO_VAR
VAR_TO_TABLE
VAR_TO_<type>
VAR_TO_VARPTR
VBX.FIRE_EVENT
VBX.GET_PROPERTY
VBX.GET_VALUE_PROPERTY

VBX.INVOKE_METHOD
VBX.SET_PROPERTY
VBX.SET_VALUE_PROPERTY

W
WEB.SHOW_DOCUMENT
WHERE_DISPLAY
WRITE_IMAGE_FILE
WRITE_SOUND_FILE

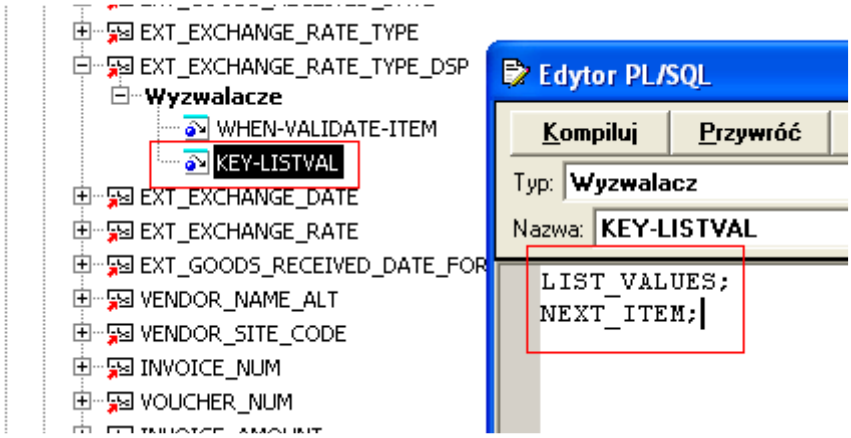
X

Y

Z

Jak dodać własną obsługę przed/po standardowym zdarzeniem KEY-.

Przykład 1/ Przejście do następnego pola po wybraniu wartości z listy



Użycie DO_KEY('LIST_VALUES') zamiast LIST_VALUES spowodowałoby zapętlenie rekurencyjne.

Przykład 2/ Sprawdzenie, czy można usunąć rekord/zmienić/wstawić
Utwórz wyzwalacz KEY-DELREC/ KEY-UPDREC/ KEY-CREREC sprawdzający, czy można usunąć rekord

Nie dodaj DELETE_RECORD/UPDATE_RECORD_INSERT_RECORD w tych zdarzeniach (one są tożsame z ON-DELETE, ON-UPDATE, ON-INSERT)

Inne podejście: zastosowanie wyzwalacza z hierarchią uruchomienia „przed".

Ukrycie komunikatu „Zatwierdzono rekordów ..."

```
declare
 defMessLevel number;
begin
defMessLevel := :SYSTEM.MESSAGE_LEVEL;
:SYSTEM.MESSAGE_LEVEL := 25;
 GO_BLOCK('VARIABLE');
 EXECUTE_QUERY;
 GO_BLOCK('FIXED');
 EXECUTE_QUERY;
 GO_BLOCK('RENT');
 EXECUTE_QUERY;
 :SYSTEM.MESSAGE_LEVEL := defMessLevel;
end;
```

Jak odświeżyć zawartość formularza po zapisaniu rekordu

Czasami, zapisanie pojedynczego rekordu ma wpływ na zawartość pozostałych rekordów.
Wówczas, po zapisaniu rekordu powinna odświeżać się zawartość bloku.
Nie można jednak wykonać polecenia EXECUTE_QUERY na wyzwalaczu WHEN-VALIDATE-RECORD, WHEN-VALIDATE-ITEM.

Rozwiązanie:
1/ Dodaj parametr REFRESH_REQUIRED z domyślną wartością N

2/ Dodaj zdarzenia POST-INSERT, POST-UPDATE z kodem
:PARAMETER.REFRESH_REQUIRED := 'Y';
3/ W zdarzeniu KEY-COMMIT odśwież zawartość bloku, gdy REFRESH_REQUIRED = ‘Y’. Ustaw wartość REFRESH_REQUIRED = ‘N’:

```
declare
  procedure refreshBlock (blockName varchar2) is
   CURRENT_BLOCK varchar2(100);
   CURRENT_ITEM  varchar2(100);
    defMessLevel number;
  begin
    defMessLevel := :SYSTEM.MESSAGE_LEVEL;
    CURRENT_BLOCK := :SYSTEM.CURRENT_BLOCK;
    current_item  := :system.current_block ||'.'||:system.current_item;

    GO_BLOCK(blockName);
    :SYSTEM.MESSAGE_LEVEL := 25;
    EXECUTE_QUERY;
    :SYSTEM.MESSAGE_LEVEL := defMessLevel;

    GO_BLOCK( CURRENT_BLOCK );
    GO_ITEM( CURRENT_ITEM );
  exception
       when OTHERS then null;
    GO_BLOCK( CURRENT_BLOCK );
    GO_ITEM( CURRENT_ITEM );
  end;

begin
  APP_STANDARD.EVENT('KEY-COMMIT');

  if :PARAMETER.REFRESH_REQUIRED = 'Y' then
       refreshBlock('XXAP_VAT_CHANGES');
       :PARAMETER.REFRESH_REQUIRED := 'N';
    fnd_message.debug('Wprowadzono pomyślnie nowe stawki podatków VAT dla linii o tym samym
kodzie podatku');
  end if;

end;
```

Przykład formularza: XXAPVATCHANGES

Filtr w postaci checkbox

1. Połóż checkbox na formularzu.
2. Oprogramuj zdarzenia WHEN-CHECKBOX-CHANGED i PRE-QUERY

WHEN-CHECKBOX-CHANGED

GO_BLOCK('XX_WF_ROUTING_RULES_V');
EXECUTE_QUERY;

PRE-QUERY

```
DECLARE
    PROCEDURE setDefaultWhere (block_name varchar2, custom_where      VARCHAR2) IS
      oryginal_where VARCHAR2(5000);
    BEGIN
      oryginal_where := get_block_property(block_name, DEFAULT_WHERE);

      IF INSTR(oryginal_where, '/*CUSTOM-WHERE-START*/') > 0 THEN
        oryginal_where := SUBSTR(oryginal_where, 1, INSTR(oryginal_where, '/*CUSTOM-WHERE-
START*/')-1);
      END IF;

      oryginal_where := NVL(oryginal_where,'0=0') || ' /*CUSTOM-WHERE-START*/ AND ' ||
NVL(custom_where, '0=0') || ' /*CUSTOM-WHERE-END*/';
      set_block_property(block_name, DEFAULT_WHERE, oryginal_where);
    END;
BEGIN
  IF Xxmsz_Tools.YNToBool ( NVL(:buttons.SHOW_CURRENT_ONLY, 'T' ) ) THEN
```

```
       setDefaultWhere ( 'XX_WF_ROUTING_RULES_V', 'SYSDATE BETWEEN NVL(BEGIN_DATE,SYSDATE) AND
NVL(END_DATE,SYSDATE)' );
  ELSE
         setDefaultWhere ( 'XX_WF_ROUTING_RULES_V', NULL );
  END IF;
END;
```

ANALIZA WARTOŚCI W BLOKU

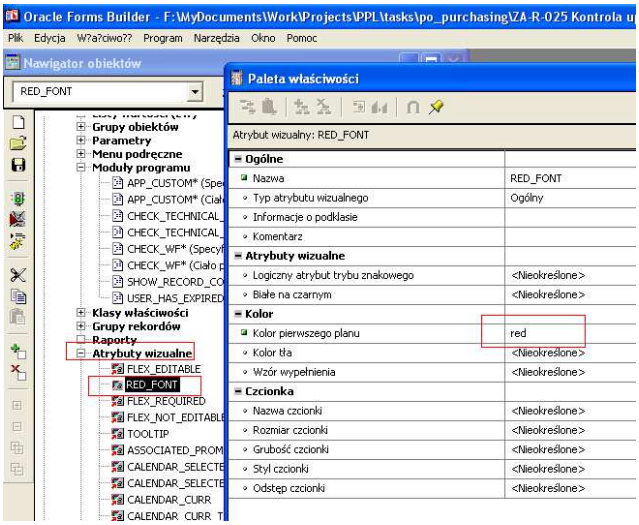
FUNCTIONALITY NO LONGER AVAILABLE IN FORMS 10G (EBS 12R)

Czasami podczas wprowadzania danych trzeba sprawdzać rekord z innymi rekordami w bloku.
Np. gdy suma wszystkich rekordów nie może przekroczyć określonej wartości, nie może być mniejsza niż 0 itd.
Gdy niemożliwe jest „chodzenie” po bloku (nie można zmienić bieżącego rekordu na inny) – użyj kodu przedstawionego poniżej.

```
DECLARE
   TYPE type_rec IS RECORD(
      PO_LINE_LOCATION_ID     XXPO_RECEIVE_REPORTS_LINES_ALL.PO_LINE_LOCATION_ID%TYPE
                    ,ILOSC                      XXPO_RECEIVE_REPORTS_LINES_ALL.ILOSC%TYPE
           );

                  TYPE type_tab IS TABLE OF type_rec INDEX BY BINARY_INTEGER;

    data_block type_tab;
    item_block ITEMS_IN_BLOCK;
    SUMA_ILOSCI NUMBER;

BEGIN
  SUMA_ILOSCI := 0;

  item_block(1):= 'PO_LINE_LOCATION_ID';
  item_block(2):= 'ILOSC';

  TABLE_FROM_BLOCK(data_block, 'XXPO_RECEIVE_REPORTS_LINES_ALL', 1, ALL_ROWS, item_block);
  IF data_block.COUNT > 0 THEN
    FOR i IN data_block.FIRST .. data_block.LAST LOOP
      IF NAME_IN('XXPO_RECEIVE_REPORTS_LINES_ALL.PO_LINE_LOCATION_ID') = data_block(i).PO_LINE_LOCATION_ID
         AND i <> :system.CURSOR_RECORD THEN
         SUMA_ILOSCI := SUMA_ILOSCI + data_block(i).ilosc;
      END IF;
    END LOOP;
  END IF;

  fnd_message.set_string('SUMA_ILOSCI lini o okreslonym ID nie liczac biezacego rekordu to ' || SUMA_ILOSCI);
  fnd_message.show;
END;
```

DYNAMICZNE USTAWIANIE KOLORU ELEMENTU

```
IF ...
  THEN  VA := 'RED_FONT';
  ELSE  VA := NULL;
END IF;
SET_ITEM_INSTANCE_PROPERTY('XX_WF_ROUTING_RULES_V.ROLE_DSP',CURRENT_RECORD, VISUAL_ATTRIBUTE, VA);

Obsłuż zdarzenia POST-QUERY, opcjonalnie WHEN-VALIDATE-ITEM, WHEN-VALIDATE-RECORD
```



WŁAŚCIWOŚĆ „ENABLED”

Ostrożnie zmieniaj właściwość ENABLED. FormsSerwer gubi się (tzn. przestaje działać nawigacja myszą), gdy np. wykonasz przejście do wyłączonego elementu.

Zamiast ENABLED użyj lepiej innych właściwości
ZAMIAST:
set_item_property('PO_REQ_HDR.SEGMENT1', ENABLED, PROPERTY_FALSE); --2008.11.03 MSz
WYKONAJ:
set_item_property('PO_REQ_HDR.SEGMENT1', required, PROPERTY_FALSE);
set_item_property('PO_REQ_HDR.SEGMENT1', insert_allowed, PROPERTY_FALSE);
set_item_property('PO_REQ_HDR.SEGMENT1', update_allowed, PROPERTY_FALSE);

```
        IF :SYSTEM.CURRENT_BLOCK = 'QP_LINES' THEN
          IF (P_EVENT = 'WHEN-VALIDATE-ITEM' AND :SYSTEM.CURRENT_ITEM = 'TYPE_CODE_DSP') OR P_EVENT = 'PRE-RECORD'
          THEN
            IF :QP_LINES.LIST_LINE_TYPE_CODE = 'PBH' THEN
                set_item_property('RENT.COMMISION_ORDER_TYPE' , ENABLED        , PROPERTY_TRUE  );
                --ENABLED=FALSE wyłącza również właściwości NAVIGABLE i UPATE ALLOWED, dlatego teraz trzeba je włączyć
                set_item_property('RENT.COMMISION_ORDER_TYPE' , REQUIRED       , PROPERTY_TRUE  );
                set_item_property('RENT.COMMISION_ORDER_TYPE' , NAVIGABLE      , PROPERTY_TRUE  );
                set_item_property('RENT.COMMISION_ORDER_TYPE' , UPDATE_ALLOWED , PROPERTY_TRUE  );
            ELSE -- = PLL
                If :system.cursor_item = 'RENT.COMMISION_ORDER_TYPE' then go_item(>RENT.NEXT_ITEM_HERE<); end if;
                -- nie można wyłączyć bieżącego elementu
                set_item_property('RENT.COMMISION_ORDER_TYPE' , REQUIRED      , PROPERTY_FALSE  );
                set_item_property('RENT.COMMISION_ORDER_TYPE'       , ENABLED        , PROPERTY_FALSE  );           END IF;
            END IF;
        END IF;


SET_ITEM_PROPERTY('QP_LINES.PROGI_CENOWE'           , ENABLED        , PROPERTY_TRUE  ); - zaznacza tylko bieżący rekord
SET_ITEM_PROPERTY('QP_LINES.OPERAND'             , ENABLED        , PROPERTY_TRUE  ); - zaznacza całą kolumnę
```

WSTAWIANIE REKORDÓW DO BLOKU Z POZIOMU KODU

```
PROCEDURE INSERT_RECORDS IS
-- PROCEDURA UżYWA POLCENIA GO_BLOCK, KTÓRE NALEżY DO GRUPY "RESTRICTED"
-- OZNACZA TO, żE MOżE BYĆ WYWOŁYWANA TYLKO PRZEZ WYZWALACZE Z KATEGORII:
-- KEY-, WHEN-NEW-, NIEKTÓRE WHEN- (NP. WHEN-BUTTON-PRESSED )
-- NIE UDA SIĘ URUCHOMIENIE TEJ PROCEDURY W WYZWALACZACH TYPU: ON- PRE- POST-.

-- ABY DOSTOSOWAC PRZYKLAD DO WLASNYCH POTRZEB ZMIEN FRAGMENTY OZNACZONE ZA POMOCA <--TU ZMIEN
          INSERT_FLAG VARCHAR2(10);
          BLOCK_NAME  VARCHAR2(50);
          CURRENT_BLOCK_NAME VARCHAR2(50);
          CURRENT_ITEM_NAME  VARCHAR2(50);
BEGIN
          BLOCK_NAME := 'ROL_PLA'; --<--TU ZMIEN

          CURRENT_BLOCK_NAME := :SYSTEM.CURRENT_BLOCK;
          CURRENT_ITEM_NAME  := :SYSTEM.CURRENT_ITEM;

  GO_BLOCK(BLOCK_NAME);

  IF NOT FORM_SUCCESS THEN
          RETURN;
  END IF;

  INSERT_FLAG := GET_BLOCK_PROPERTY(BLOCK_NAME, INSERT_ALLOWED);
```

```
  --ABY WSTAWIENIE POWIODŁO SIĘ, USTAW MOŻLIWOŚĆ WSTAWIANIA REKORDÓW DLA BLOKU
  SET_BLOCK_PROPERTY(BLOCK_NAME, INSERT_ALLOWED, PROPERTY_TRUE);

  IF :SYSTEM.RECORD_STATUS IN ('NEW','INSERT','CHANGED') THEN
    CLEAR_BLOCK(ASK_COMMIT    );
  END IF;

  CLEAR_BLOCK; -- NIE UŻYWAJ INSERT_RECORD. ALTERNATYWNIE MOŻESZ UŻYĆ POLECENIA DO_KEY('CREATE_RECORD')
  :ROL_PLA.PLA_ID := 1;
  DO_KEY('CREATE_RECORD');
  :ROL_PLA.PLA_ID := 2;
  DO_KEY('CREATE_RECORD');
  :ROL_PLA.PLA_ID := 21;
  DO_KEY('CREATE_RECORD');
  :ROL_PLA.PLA_ID := 41;
  --DO_KEY('COMMIT_FORM'); -- EWENTUALNIE <--TU ZMIEN

  -- PRZYWRÓĆ STAN BLOKU ZPRZED OPERACJI
  IF INSERT_FLAG = 'TRUE' THEN
         SET_BLOCK_PROPERTY(BLOCK_NAME, INSERT_ALLOWED, PROPERTY_TRUE);
  ELSE
         SET_BLOCK_PROPERTY(BLOCK_NAME, INSERT_ALLOWED, PROPERTY_FALSE);
  END IF;
         GO_BLOCK ( CURRENT_BLOCK_NAME );
         GO_ITEM  ( CURRENT_ITEM_NAME  );
EXCEPTION
  WHEN OTHERS THEN
    -- W RAZIE BŁĘDU PRZYWRÓĆ STAN BLOKU ZPRZED OPERACJI...
    IF INSERT_FLAG = 'TRUE' THEN
         SET_BLOCK_PROPERTY(BLOCK_NAME, INSERT_ALLOWED, PROPERTY_TRUE);
    ELSE
         SET_BLOCK_PROPERTY(BLOCK_NAME, INSERT_ALLOWED, PROPERTY_FALSE);
    END IF;
          GO_BLOCK ( CURRENT_BLOCK_NAME );
          GO_ITEM  ( CURRENT_ITEM_NAME  );
    -- ...I WYWOŁAJ STANDARDOWĄ OBSŁUGĘ WYJĄTKU
    RAISE;
END;
```

DYNAMICZNA WARTOŚĆ DOMYŚLNA

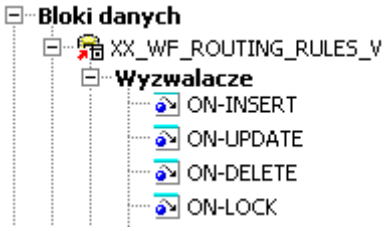
```
Zdarzenie 'WHEN-NEW-RECORD-INSTANCE'

-- wartoœæ domyœlna w polu NUM
declare
        next_num integer;
        curr_item varchar2(100);

  function getnextnum return number is
     type type_rec is record(
         us_sciezka_zatwierdzania_id us_sciezka_zatwierdzania.us_sciezka_zatwierdzania_id%type
                ,num                          us_sciezka_zatwierdzania.process_order%type
          );

               type type_tab is table of type_rec index by binary_integer;

      data_block type_tab;
               item_block items_in_block;
               max_num number := 0;
               max_num_db number := 0;
  begin
    max_num := 0;

    item_block(1):= 'US_SCIEZKA_ZATWIERDZANIA_ID';
    item_block(2):= 'NUM';

    table_from_block(data_block, :system.current_block, 1, all_rows, item_block);
       if data_block.count > 0 then
         for i in data_block.first .. data_block.last loop
                     max_num := greatest (max_num, nvl(data_block(i).num,0));
      end loop;
        end if;

    select nvl(max(process_order),0 )
      into max_num_db
      from us_sciezka_zatwierdzania
     where type = replace( :system.current_block , 'SCIEZKA_', 'US_')
       and umowa_id = :header.umowa_id;

    --fnd_message.debug('max_num=' || max_num);
    --fnd_message.debug('max_num_db=' || max_num_db);
```

```
    return  greatest(max_num, max_num_db) +10 ;
  end;

begin
  if :system.mode <> 'NORMAL' then return; end if;
  if :system.current_block not in
('SCIEZKA_ZAT','SCIEZKA_ZAT_OST','SCIEZKA_KONSULT','SCIEZKA_FIN_PRAW') then return;  end if;
       curr_item := :system.current_block || '.NUM';

       if :header.umowa_id is null then return; end if;
       if name_in (curr_item) is not null then return; end if;
       if :system.record_status <> 'NEW' then return; end if;

  next_num :=  getNextNum;
  copy(next_num, curr_item );
  SET_RECORD_PROPERTY(:SYSTEM.TRIGGER_RECORD, :system.current_block, STATUS , NEW_STATUS);
end;
```

ON-INSERT i COŚ JESZCZE

Czasami oprócz wstawienia, aktualizacji Lub usunięcia rekordu trzeba wykonwać jeszcze jakąś czynność dodatkową (np. zaktualizowanie czegoś w innym module). Zrób tak:

Nadpisz standardowe zdarzenia...

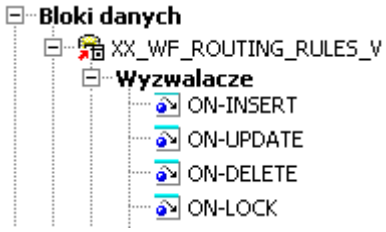


...w następujący sposób:

```
if p_event = 'ON-INSERT' then
 oe_on_insert; -- własna procedura
 insert_record; -- standardowa procedura
 if not form_success then raise form_trigger_failure; end if;

elsif p_event = 'ON-UPDATE' then
 oe_on_update;
 update_record;
 if not form_success then raise form_trigger_failure; end if;

elsif p_event = 'ON-DELETE' then
 oe_on_delete;
 delete_record;
 if not form_success then raise form_trigger_failure; end if;
```

FORMATKA OPARTA NA WIDOKU

Możesz oprzeć formatkę na widoku.
Wówczas obsłuż następujące zdarzenia na poziomie bloku:

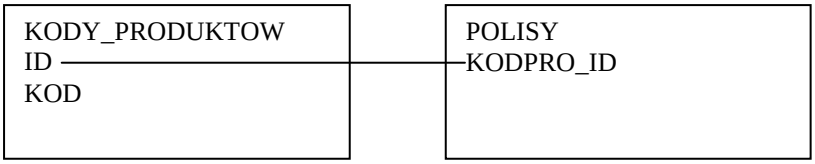


W ON-LOCK możesz wpisać po prostu NULL lub kod blokujący rekord (poniżej przykład)

```
declare
      cursor c_row is
        select *
         from US_UMOWY_NIE_NAJMU_V
           where agreement_id = :UNN_DETAILS.AGREEMENT_ID
           for update nowait;
          v_row c_row%ROWTYPE;
begin
  open c_row;
  fetch c_row into v_row;
  if c_row%NOTFOUND then
    fnd_message.set_string('Błąd ewenętrzny. NO_DATA_FOUND podczas próby zarezerwowania
rekordu');
    fnd_message.error;
    close c_row;
    raise no_data_found;
  else
    close c_row;
  end if;
exception
  when others then
    if SQLCODE = -54 then
      fnd_message.set_string('Nie mo¿na zarezerwowaæ rekordu.'||SQLERRM);
      fnd_message.error;
    ELSE
      fnd_message.set_string('Błąd ewenętrzny. Podczas próby zarezerwowania rekordu zaszło
zdarzenie :' || SQLERRM);
      fnd_message.error;
    end if;
    raise form_trigger_failure;
end;
```

Uwaga:
Właściwość ENFORCE_PRIMARY_KEY dla bloku musi być ustawiona na FALSE
SET_BLOCK_PROPERTY('UNN_DETAILS',ENFORCE_PRIMARY_KEY, PROPERTY_FALSE);

LISTA WARTOŚCI



C

1. UTWÓRZ FORMANTY DO WYŚWIETLANIA PÓL

   Pole tekstowe: POLISY.KODPRO_ID (niewidoczny)
   Pole tekstowe: POLISY.L_KOD (widoczny)
   Przycisk: z akcją go_item('POLISY.L_KOD'); do_key('List_values');

2. Utworzyć LOV za pomocą kreatora

   - UTWORZYĆ GRUPĘ REKORDÓW GR$KP
     ZAPYTANIE DLA GRUPY REKORDOW = SELECT ID,KOD FROM KODY_PRODUKTOW

   - UTWORZYĆ LISTĘ WARTOŚCI LV$KP
     GRUPA REKORDOW = GR$KP
     WŁAŚCIWOŚCI ODWZOROWANIA KOLUMN =
     ID = POLISY.KODPRO_ID
     KOD = POLISY.L_KOD

     POLISY.KODPRO_ID: LISTA_WARTOSCI.LISTAWARTOŚCI ZOSTANIE USTAWIONA AUTOMATYCZNIE

   - ŻEBY ZMIENIĆ DOMYŚLNE ROZMIARY OKNA Z LISTĄ WARTOŚCI I SZEROKOŚCI KOLUMN
     LISTA WARTOŚCI.FIZYCZNE...
     LISTA WARTOŚCI.FUNKCJONALNE.WŁAŚCIWOŚCI ZOBRAZOWANIA KOLUMN.WYSWIATLANA SZEROKOSC

3. Dodaj zdarzenie WHEN-VALIDATE-ITEM dla pola polisy.l_kod

```
IF     :polisy.l_kod IS NULL and :polisy.kod_pro_id is not null
THEN
   :polisy.kod_pro_id := NULL;
END IF;
```

+ ewentualna obsługa walidacji jeżeli wyłączysz walidację przez listę.

4. (OMIŃ TEN KROK DLA WARIANTU Z KOLUMNĄ Z BAZY DANYCH) ŻEBY ELEMENT NIEBAZOWY ODŚWIEŻAŁ SIĘ, DODAJ:

```
POLISY.POST-QUERY:
BEGIN
 SELECT KOD INTO :POLISY.L_KOD
 FROM KODY_PRODUKTOW
 WHERE ID = :POLISY.KODPRO_ID;
END;
SET_RECORD_PROPERTY(:SYSTEM.TRIGGER_RECORD, :system.current_block, STATUS , QUERY_STATUS);
-- SET_RECORD_PROPERTY zapewnia to, że rekord nie będzie otrzymywał statusu „zmieniony”.
```

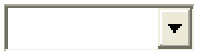
UWAGA:

LOV zbudowany w oparciu o pole niebazodanowe posiada wady:

1. Zapytania nie będą działały dla pola niebazodanowego (trzeba ręcznie oprogramować zdarzenie PRE-QUERY i poprzez ręczne doklejanie warunku do DEFAULT_WHERE – przykładowy kod poniżej – napisany tekstem ukrytym)
2. Nie będzie można sortować danych w bloku wg tego pola

Aby tego uniknąć: 1. zbuduj blok na widoku **(rekomendowane)** lub 2. dodaj do tabeli kolumnę zdenormalizowaną. Zamiast tego obsłuż zdarzenie when-validate-item.

POLE KOMBI – LISTA DYNAMICZNA



1. UTWORZYĆ ELEMENT KODPRO_ID
   OGÓLNE.TYP ELEMENTU = LISTA
   FUNKCJONALNE.STYL LISTY=LISTA ROZWIJANA

2. ZDEFINIOWAĆ FUNKCJĘ ŁADUJĄCĄ WARTOŚCI:

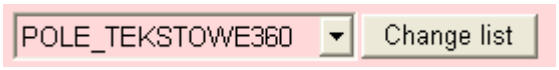
```
PROCEDURE LOAD_LOOKUPS IS
  ora_err NUMBER;
  rg_id   RECORDGROUP;
BEGIN
  rg_id := CREATE_GROUP_FROM_QUERY('CGLL$POP_LIST','SELECT NAPIS_WYSWIETLANY
E,WARTOSC CG$VALCOLALIAS FROM WARTOSCI_LIST');
  -- Z DEFINICJI GRUPA MUSI SKŁADAĆ SIĘ Z DWÓCH KOLUMN: ETYKIETA, WARTOŚĆ
  ora_err := POPULATE_GROUP(rg_id);

  IF ora_err > 0 THEN
       MESSAGE('BŁĄD PRZY ŁADOWANIU LOOKUPA WARTOŚCIAMI ORA-
0'||TO_CHAR(ora_err));
       MESSAGE('BŁĄD PRZY ŁADOWANIU LOOKUPA WARTOŚCIAMI ORA-
0'||TO_CHAR(ora_err)) ;
       RAISE FORM_TRIGGER_FAILURE;
  END IF;

  POPULATE_LIST('DOKUMENTY.POUFNOSC', rg_id);
  DELETE_GROUP(rg_id);
END;
```

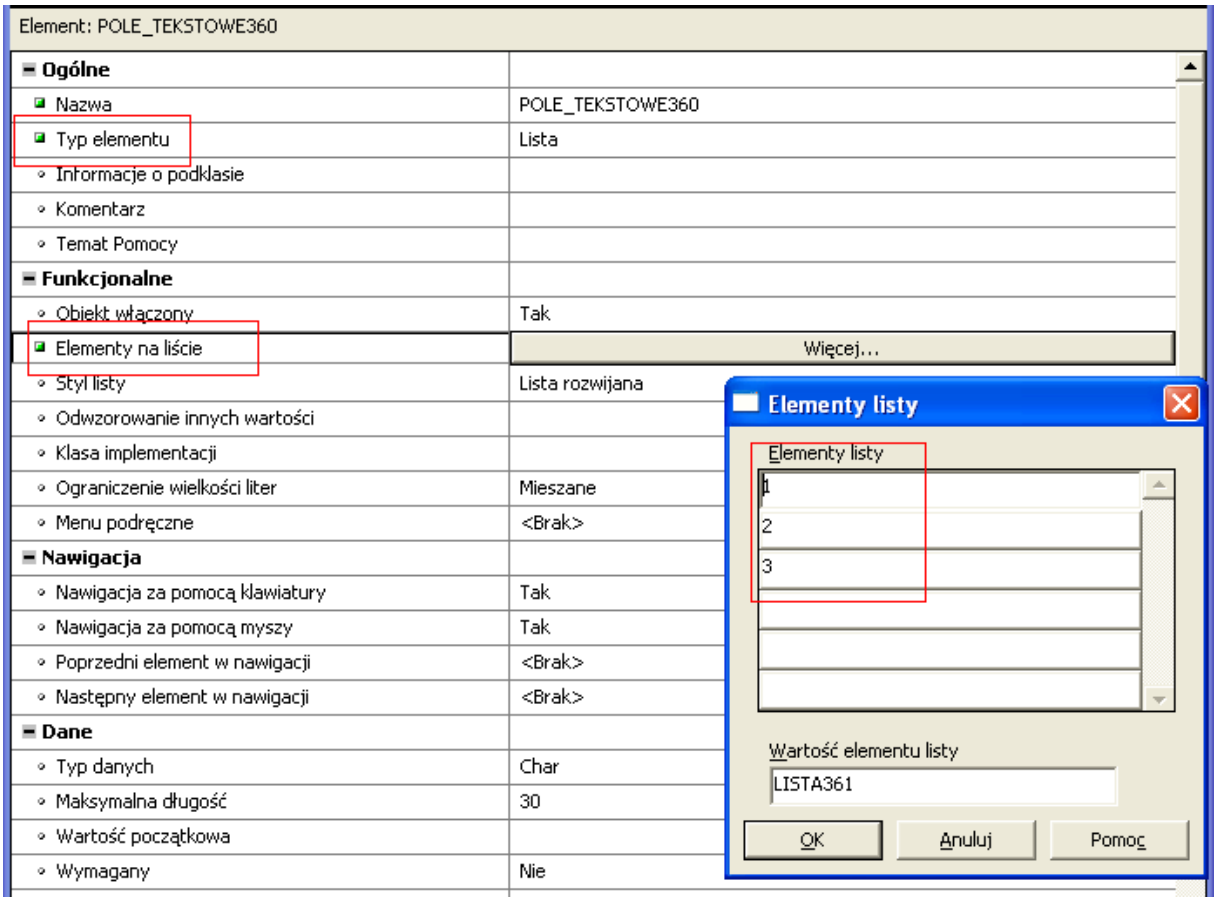
3. WYWOŁAĆ FUNKCJĘ W WHEN_NEW_FORM_INSTANCE I KEY-CLRFORM

POLE COMBI – LISTA STATYCZNA

1. UTWORZYĆ ELEMENT RODZAJ
   TYP ELEMENTU = LISTA
   ELEMENTY NA LIŚCIE – TU WPISZ ELEMENTY
   STYL LISTY = POLE KOMBI
   ELEMENT BAZY DANYCH = TAK

Modyfikacja statycznej listy wartości



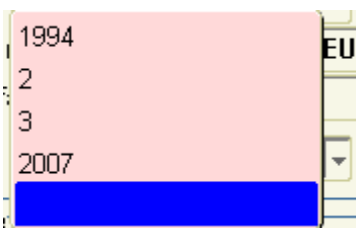


Kod dla przycisku „change”:

```
Delete_List_Element('POLE_TEKSTOWE360',1);
 Add_List_Element('POLE_TEKSTOWE360', 1, '1994val', '1994code');
 Add_List_Element('POLE_TEKSTOWE360', 4, '2007val', '2007code');
```

Przed naciśnięciem przycisku:

Po naciśnięciu przycisku:



NAWIGACJA

KOLEJNOŚĆ NAWIGACJI JEST WYZNACZONA PRZEZ KOLEJNOŚĆ BLOKÓW I ELEMENTÓW NA LIŚCIE.

PODŁĄCZANIE SEKWENCJI (STANDARD DESIGNERA)

OBSŁUŻYĆ ZDARZENIE PRE-INSERT DLA BLOKU DANYCH, NP.

```
IF (name_in('LISDES.LISDES_ID') IS NULL) THEN
   SELECT LISDES_SEQ.NEXTVAL INTO :LISDES.LISDES_ID FROM DUAL;
END IF;
```

WYWOŁYWANIE FORMULARZA

OPEN_FORM – WYWOŁANIA NIEMODALNE
NEW_FORM – ZAMKNIĘCIE BIEZACEGO FORMULARZA I WYWOŁANIE NOWEGO
CALL_FORM – WYWOŁANIE MODALNE

OPEN_FORM(form_name, activate_mode,session_mode, plist)
New_form(form_name,rollback_mode,query_mode,plist)
CALL_FORM(form_name,display,switch_menu,query_mode,plist)

NAWIGOWANIE POMIĘDZY FORMULARZAMI

GO_FORM
NEXT_FORM
CLOSE_FORM
EXIT_FORM

UKRYWANIE/ POJAWIANIE SIE PRZYCISKU

```
SET_ITEM_PROPERTY('MAIN.OPEN_LOG',VISIBLE,PROPERTY_FALSE);
SET_ITEM_PROPERTY(' MAIN.OPEN_LOG ',ENABLED,PROPERTY_FALSE);

SET_ITEM_PROPERTY('MAIN.OPEN_LOG',VISIBLE,PROPERTY_TRUE);
SET_ITEM_PROPERTY(' MAIN.OPEN_LOG ',ENABLED,PROPERTY_TRUE);
```

ZMIANA ETYKIETY ELEMENTU

```
SET_ITEM_PROPERTY('MAIN.CHB_EKSPORT_SIEROTY',LABEL,'TEKST');
```

TREE

1. UTWORZYĆ KOMPONENT TREE W OSOBNYM, JEDNOWIERSZOWYM BLOKU
2. WYPEŁNIĆ DRZEWKO NP. W TEN SPOSÓB

```
PROCEDURE FILL_TREE IS
grupa_ksia  recordgroup;
t_init      groupcolumn;
t_level     groupcolumn;
t_label     groupcolumn;
t_icon      groupcolumn;
t_value     groupcolumn;
--ile_tree    number;
htree       item;
l_ikona     number;

t_idx       number;
BEGIN
        grupa_ksia := find_group('GR_KSIA');
        if not id_null(grupa_ksia) then
                delete_group(grupa_ksia);
        end if;

        grupa_ksia := create_group('GR_KSIA');
        t_init := add_group_column(grupa_ksia, 'init', number_column);
        t_level := add_group_column(grupa_ksia, 'level', number_column);
        t_label := add_group_column(grupa_ksia, 'label', char_column, 90);
        t_icon := add_group_column(grupa_ksia, 'icon', char_column, 20);
        t_value := add_group_column(grupa_ksia, 'name', char_column, 10);

        t_idx := 1;
        l_ikona := 0;
        for REC in (SELECT  NAZWA  FROM PARAGRAFY) loop
                add_group_row(grupa_ksia, t_idx);
          set_group_number_cell(t_init, t_idx, 1);
          set_group_number_cell(t_level, t_idx, 1);--adrrec.level);
        set_group_char_cell(t_label, t_idx, REC.NAZWA);
        if l_ikona = 1 THEN --adrrec.level then
                                set_group_char_cell(t_icon, t_idx, 'docs' );
        else  set_group_char_cell(t_icon, t_idx, 'open' );       end if;
        set_group_char_cell(t_value, t_idx, t_idx);
          t_idx := t_idx + 1;
          l_ikona := 1; --adrrec.level;
        end loop;

        htree := find_item(<'BLOCK_TREE_TO.TREE'>);
        ftree.set_tree_property(htree, ftree.record_group, grupa_ksia);
        delete_group(grupa_ksia);
        --ile_tree := ftree.get_tree_property(htree, ftree.node_count);
END;
```

3. DODAĆ ZDARZENIE WHEN-TREE-NODE-SELECTED

```
DECLARE
  htree        ITEM;
  node_value   VARCHAR2(100);
  NODE                                      FTREE.NODE;
BEGIN
  htree := Find_Item(<'BLOCK_TREE_TO.TREE'>);
  node  := :SYSTEM.TRIGGER_NODE;
```

```
  node_value := Ftree.Get_Tree_Node_Property(htree, node, Ftree.NODE_VALUE);
MESSAGE(NODE_VALUE);
END;
```

NODE_DEPTH Returns the nesting level of the hierarchical tree node.
NODE_LABEL Returns the label
NODE_ICON Returns the icon name
NODE_VALUE Returns the value of the hierarchical tree node.

DYNAMICZNA LISTA LOV

1. UTWORZYĆ GRUPE REKORDOW GR
   ZAPYTANIE DLA GRUPY REKORDOW = select 'KOLUMNA' KOLUMNA, 'typ' TYP_DANYCH from dual
2. UTWORZYĆ LISTĘ WARTOŚCI(LV) LV
   GRUPA REKORDOW = GR
   WŁAŚCIWOŚCI ODWZOROWANIA KOLUMN=KOLUMNA = BLOK.KOLUMNA=NAGŁÓWEK
3. UTWORZYĆ PRZYCISK Z AKCJĄ WHEN-BUTTON-PRESSED

```
DECLARE
  rg_name  VARCHAR2(40) := 'COLQ_TABELE';
  rg_id    RecordGroup;
  errcode NUMBER;
  lov_id      LOV;
BEGIN
 lov_id:= Find_LOV('LV');
 rg_id := Find_Group( rg_name);
 IF Id_Null(rg_id) THEN
   rg_id := Create_Group_From_Query( rg_name, 'select COLUMN_NAME KOLUMNA,DATA_TYPE
TYP_DANYCH from user_tab_columns where TABLE_NAME='''||:ULIA.NAZMIA||''' ORDER BY 1');
 END IF;
-- NAZWY KOLUMN MUSZĄ BYĆ ZGODNE Z ZADEKLAROWANYMI W GRUPIE REKORDOW
set_lov_property(lov_id ,TITLE,'Kolumny z tabeli '||UPPER(:ULIA.NAZMIA));
set_lov_property(lov_id ,GROUP_NAME,rg_name);

--AUTO_REFRESH  Specifies whether Form Builder re-executes the query each time the LOV is invoked.
--GROUP_NAME  Specifies the record group with which the LOV is associated.
--LOV_SIZE  Specifies a width, height pair indicating the size of the LOV.
--POSITION  Specifies an x, y pair indicating the position of the LOV.
--TITLE  Specifies the title of the LOV. Overrides the value specified in the Form Builder unless the
property value is NULL.

go_item('BLOK.KOLUMNA');
do_key ( 'List_Values' );
delete_group(rg_id);
END;
```

OBRAZY W BAZIE DANYCH

1. PRZECHOWYWAĆ W POLU O TYPIE LOG RAW
2. UTWORZYĆ ELEMENT TYPU OBRAZ **ZDJ.OBRAZEK**
3. UTWORZYĆ ELEMENT TYPU PRZYCISK „WCZYTAJ”
   WHEN-BUTTON-PRESSED
   DECLARE
   FILE_NAME_TO_READ VARCHAR2(500);
   BEGIN
   FILE_NAME_TO_READ := GET_FILE_NAME(directory_name=> FILE_NAME_TO_READ,
   File_Filter=> 'Pliki tekstowe (*.*)|*.*|');
   CGIM$LOAD(FILE_NAME_TO_READ,'**ZDJ.OBRAZEK**');
   EXCEPTION
   WHEN OTHERS THEN
   CGTE$OTHER_EXCEPTIONS;
   END;
1. UTWORZYĆ ELEMENT TYPU PRZYCISK „ZACHOWAJ”
   WHEN-BUTTON-PRESSED
   WRITE_IMAGE_FILE('C:\TEST.BMP','BMP','ZDJ.OBRAZEK');

```
PROCEDURE CGIM$LOAD(
  P_iname IN VARCHAR2
 ,P_iitem IN VARCHAR2) IS
BEGIN
 IF :SYSTEM.MODE = 'NORMAL' THEN
  BEGIN
   CGIM$READ_IMAGE_FILE(
    P_iname
   ,P_iitem);
  EXCEPTION
   WHEN OTHERS THEN
    CGTE$OTHER_EXCEPTIONS;
  END;
 END IF;
END;

procedure CGIM$READ_IMAGE_FILE(iname IN varchar2, iitem in varchar2) is
mypic item;
itype char(4) := upper(substr(iname,greatest((length(iname)-3),1)));
begin
 if (iname IS NOT NULL)
 then
  mypic := find_item(iitem);
  if (get_item_property(mypic,UPDATEABLE) = 'FALSE'
     and
     NAME_IN('system.record_status') in ('CHANGED','QUERY')
     and
     NAME_IN('system.mode') = 'NORMAL')
  then
     message('ERROR: Image not updateable');
     raise FORM_TRIGGER_FAILURE;
  else
     if (itype not in ('TIFF','JFIF','PICT'))
     then itype := upper(substr(iname,greatest((length(iname)-2),1)));
     end if;
     if (itype = 'TIF')
     then itype := 'TIFF';
     end if;
     read_image_file(iname, itype, mypic);
```

```
    end if;
  end if;
end;
```

INNE POLECENIA DO OBRÓBKI OBRAZU:

IMAGE_ZOOM

IMAGE_SCROLL

DZWIĘKI W BAZIE DANYCH

1. PRZECHOWYWAĆ W POLU O TYPIE LOG RAW
2. UTWORZYĆ ELEMENT TYPU DZWIĘK **DZW.DZWIEK**
3. UTWORZYĆ ELEMENT TYPU PRZYCISK **WCZYTAJ**

```
WHEN-BUTTON-PRESSED
DECLARE
    FILE_NAME_TO_READ VARCHAR2(500);
BEGIN
    FILE_NAME_TO_READ := GET_FILE_NAME(directory_name=> FILE_NAME_TO_READ,
File_Filter=> 'Wszystkie pliki (*.*)|*.*|');
  read_sound_file(FILE_NAME_TO_READ, 'WAVE','DZW.DZWIEK');
END;
```

4. UTWORZYĆ ELEMENT TYPU PRZYCISK **ZACHOWAJ**

```
WHEN-BUTTON-PRESSED
WRITE_SOUND_FILE('C:\TEST.BMP',' WAVE ', 'DZW.DZWIEK');
```

5. UTWORZYĆ ELEMENT TYPU PRZYCISK **ODTWARZAJ**

```
WHEN-BUTTON-PRESSED
PLAY_SOUND(item_name VARCHAR2);
```

WYWOŁYWANIE FORMULARZA Z PARAMETREM (STANDARD DESIGNERA)

```
/* CGNV$AI_WINDOW_RPF_POL_DET_005 */
PROCEDURE CGNV$AI_WINDOW_RPF_POL_DET_005 IS
/* Performs the action for an action item */
  pl_id PARAMLIST;
  pl_name VARCHAR2(10);
BEGIN
  pl_name :=  'temp';
  pl_id :=  get_parameter_list(pl_name);
  IF NOT ID_NULL(pl_id) THEN
    destroy_parameter_list(pl_id);
  END IF;
  pl_id := create_parameter_list(pl_name);
  IF ID_NULL(pl_id) THEN
    CG$FORM_ERRORS.PUSH(CG$FORM_ERRORS.MSGGETTEXT(16, 'Nieudane wywołanie procedury
"<p1>"'), 'E', 'OFG', 16);
    CG$FORM_ERRORS.RAISE_FAILURE;
  END IF;
  add_parameter(pl_id, 'CG$STARTUP_MODE', TEXT_PARAMETER, 'AUTO QUERY');
  add_parameter(pl_id, 'ID', TEXT_PARAMETER, to_char(:POLISY.ID));

  call_form('RPF_POL_DET', HIDE, NO_REPLACE, NO_QUERY_ONLY, NO_SHARE_LIBRARY_DATA,
pl_id);
  IF NOT form_success THEN
    CG$FORM_ERRORS.PUSH(CG$FORM_ERRORS.MSGGETTEXT(68, 'Nie można wywołać modułu',
'RPF_POL_DET'), 'E', 'OFG', 68);
    CG$FORM_ERRORS.RAISE_FAILURE;
  END IF;
END;
```

FORMULARZ WYWOŁYWANY MUSI MIEĆ ZDEFINIOWANE PARAMETRY ZGODNIE Z PRZEKAZYWANĄ LISTĄ PARAMETRÓW.

OBSŁUŻYĆ ZDARZENIE
WHEN-NEW-FORM-INSTANCJE
GO_BLOCK
INSERT_RECORD;
BLOK.POL_ID := :PARAMETER.NAZWA

ENTER_QUERY;
BLOK.POL_ID := :PARAMETER.NAZWA
EXECUTE_QUERY;

--
set_block_property('STANOWISKA',order_by,'JED_JED_ID, KOD');
set_window_property(FORMS_MDI_WINDOW, window_state, maximize);
item := find_item(varchar2)

DO_KEY

BLOCK_MENU , CLEAR_BLOCK, CLEAR_FORM, CLEAR_RECORD, COMMIT_FORM, COUNT_QUERY, CREATE_RECORD, DELETE_RECORD, DOWN, DUPLICATE_ITEM, DUPLICATE_RECORD, EDIT_TEXTITEM, ENTER, ENTER_QUERY, EXECUTE_QUERY, EXIT_FORM, HELP, LIST_VALUES, LOCK_RECORD, NEXT_BLOCK, NEXT_ITEM, NEXT_KEY, NEXT_RECORD, NEXT_SET, PREVIOUS_BLOCK, PREVIOUS_ITEM, PREVIOUS_RECORD, PRINT, SCROLL_DOWN, SCROLL_UP, UP

CHECK-RECORD

```
BLOK.WHEN-VALIDATE-RECORD
        IF NVL(POLE1,’.’)<>’1’ AND NVL(POLE2,’.’)<>’1’ THEN
                MESSAGE(...); GO_ITEM(POLE1);
                RAISE FORM_TRIGGER_FAILURE;
        END IF;
```

IMPLEMENTACJA CHECKBOX

```
ELEMENT.FUNKCJONALNE.WARTOSC ZAZNACZONEGO POLA = ...
ELEMENT.FUNKCJONALNE.WARTOSC NIE ZAZNACZONEGO POLA = ...
ELEMENT.WARTOŚĆ POCZĄTKOWA = ...
:BLOK.ELEMENT := ‘...’;
```

FUNKCJA PODSUMOWUJĄCA

```
BLOK PODSUMOWYWANY.BD ZAAWANSOWANE.PRECOMPUTE SUMMARIES PROPERTY := TAK;
BLOK INNY.SINGLE RECORD = YES
        ELEMENT (NP. LICZBA UPOSAŻONYCH)
                DANE.TYP = NUMBER
                OBLICZENIA.RODZAJ = PODSUMOWANIE
                FUNKCJA = LICZNIK
                BLOK = BLOK PODSUMOWYWANY
                ELEMENT = ELEMENT Z WARTOŚCIAMI NOT NULL!
                BAZA DANYCH.ELEMENT BAZY DANYCH = NIE
```

Zablokowanie wyjścia z bloku bez zapisania rekordu

Spędziłem na tym trywialnym problemem dobre 4 godz, aż dostałem pomoc od przyjaciela.

Zdefiniuj parametr:
```
:PARAMETER.LET_ME_POST_BLOCK := 'N';
```

W zdarzeniu KEY-COMMIT:
```
        :PARAMETER.LET_ME_POST_BLOCK := 'Y';
        APP_STANDARD.EVENT('KEY-COMMIT'); (=DO_KEY(‘COMMIT-FORM’))
```

W zdarzeniu POST-BLOCK:
```
        if :system.record_status in ('CHANGED',’NEW’)  then
         --gdyby ten warunek nie zadziałał, to użyj: :system.block_status ='CHANGED'
                if :PARAMETER.LET_ME_POST_BLOCK = 'Y' then
                        :PARAMETER.LET_ME_POST_BLOCK := 'N';
                else
                        fnd_message.set_string('Przed wyjściem z bloku musisz zapisać zmiany');
                        fnd_message.show;
                  raise form_trigger_failure;
                end if;
        end if;
```

Musisz ustawiać również :PARAMETER.LET_ME_POST_BLOCK := 'Y'; przed uruchomieniem kalendarzy I fleksów (dotyczy Oracle Applications) I możesz na wszelki wypadek ustawiać :PARAMETER.LET_ME_POST_BLOCK := 'N'; w zdarzeniu when-new-block-instance.

WYMUSZENIE WPROWADZENIA REKORDU PODRZĘDNEGO

```
FORMULARZ
        PRE-COMMIT
                IF :BLOK.LICZBA_UPOSAZONYCH = 0 THEN
                        MESSAGE('...');
                        RAISE FORM_TRIGGER_FAILURE;
                END IF;
```

UWAGA: BLOK NADRZĘDNY MUSI BYĆ Z JEDNYM REKORDEM !

WYMUSZENIE ZATWIERDZENIA REKORDU NADRZĘDNEGO

NIESPRAWDZONE
WYWOŁYWAĆ W ZDARZENIU ON_CLEAR_DETAILS

```
DECLARE
MASTER_RECORD NUMBER;
BEGIN
        VALIDATE(RECORD_SCOPE);
        IF NOT FORM_SUCCESS THEN RAISE FORM_TRIGGER_FAILURE; END IF;
        MASTER_RECORD :=
        GET_BLOCK_PROPERTY(NAME_IN('SYSTEM.MASTER_BLOCK'),CURRENT_RECORD);
        IF GET_RECORD_PROPERTY(MASTER_RECORD,
        NAME_IN('SYSTEM.MASTER_BLOCK'),STATUS) IN ('INSERT','CHANGED') THEN RAISE
        FORM_TRIGGER_FAILURE
END;
```

Zmiana wyglądu wskaźnika myszy

```
set_application_property(cursor_style,'INSERTION');
```

BUSY	Displays a GUI-specific busy symbol.
CROSSHAIR	Displays a GUI-specific crosshair symbol.
DEFAULT	Displays a GUI-specific arrow symbol.
HELP	Displays a GUI-specific help symbol.
INSERTION	Displays a GUI-specific insertion symbol.

ZEGARY (funkcjonalność nie obsługiwana przez forms serwer)

```
UTWORZYĆ ZEGAR
        CREATE_TIMER('NAZWA',3600 [MILISEKUNDY], REPEAT|NO_REPEAT);
ZDEFINIOWAĆ ZDARZENIE
        FORMULARZ
                WHEN-TIMER-EXPIRED
                        DECLARE
                                NAME VARCHAR2(30);
                        BEGIN
                                NAME := GET_APPLICATION_PROPERTY(TIMER_NAME);
                                IF NAME = '...' THEN
                                        ...
                                END IF;
                        END;

FIND_TIMER('NAZWA')
CREATE_TIMER('NAZWA',3600 [MILISEKUNDY], REPEAT|NO_REPEAT);
```

SET_TIMER(‘NAZWA’|ID, WARTOSC|NO_CHANGE, REPEAT|NO_REPEAT)
DELETE_TIMER(‘NAZWA’|ID)
GET_APPLICATION_PROPERTY(TIMER_NAME)

LOGOWANIE DO BAZY

LOGON(USER, PASSWORD, LOGON_SCREEN_ON_ERROR)
LOGON_SCREEN;
LOGOUT

STEROWANIE INTERFEJSM ZAPYTANIA

Rozpoczęcie: ENTER_QUERY,
EXECUTE_QUERY,
COUNT_QUERY,
Wymuszenie sprowadzenie wszystkich rekordów:
ENTER_QUERY(ALL_RECORDS),
EXECUTE_QUERY(ALL_RECORDS)
Zakończenie: ABORT_QUERY, CLEAR_QUERY

Sprawdzanie stanu formularza:
:SYSTEM.MODE = {NORMAL, ENTER-QUERY, QUERY – w trakcie, np. w wyzwalaczu POST-QUERY}

STATUSY

{GET/SET}_{RECORD/BLOCK/FORM}_PROPERTY

:SYSTEM.RECORD_STATUS

NEW	NOWY, NIE ZMIENIONY
INSERT	NOWY I ZMIENIONY
QUERY	ISTNIEJĄCY, NIE ZMIENIONY
CHANGED	ISTNIEJĄCY, ZMIENIONY

:SYSTEM.BLOCK_STATUS

NEW	WSZYSTKIE REKORDY MAJĄ STATUS NEW LUB INSERT
QUERY	
CHANGED	

:SYSTEM.FORM_STATUS

NEW	WSZYSTKIE BLOKI MAJĄ STATUS NEW
QUERY	WSZYSTKIE BLOKI MAJĄ STATUSY NEW I QUERY
CHANGED	ISTNIEJE BLOK O STATUSIE CHANGED

Możesz zmienić status rekordu za pomocą polecenia
SET_RECORD_PROPERTY (:system.trigger_record, 'xxaplines', status, QUERY_STATUS);

Problem: nie działa zdarzenie WHEN-VALIDATE-ITEM

Zdarzenie walidacji nie jest uruchamiane przy rzejściu do elementu nienawidowalnego myszą (np. przycisku).
Obejście: button.on_enter = begin enter; end;

WYWOŁYWANIE OKIENKA DIALOGOWEGO Z PYTANIEM T/N

UTWORZYĆ ALERT:

NAZWA = CFG_INFORMATION_M
ETYKIETA PRZYCISKU1 = TAK
ETYKIETA PRZYCISKU2 = NIE
DOMYŚLNY ALERT PRZYCISKU = PRZYCISK2

```
FUNCTION QUESTION_INFO( M IN VARCHAR2) RETURN BOOLEAN IS
 al alert:=Find_Alert('CFG_INFORMATION_M');
 bt number;
BEGIN
 Set_Alert_Property(al,ALERT_MESSAGE_TEXT,M);
 bt:=Show_alert(al);
 IF bt=Alert_Button1 THEN
  RETURN TRUE;
 ELSE
        RETURN FALSE;
 END IF;
END;
```

STANDARDOWE MASKI FORMS I REPORTS

DD-MM-YYYY HH24-MI-SS
99”-„999
FM999G999G990D00 (FORMS)
FMNNNGNNNGNN0D00 (REPORTS)

BAZA

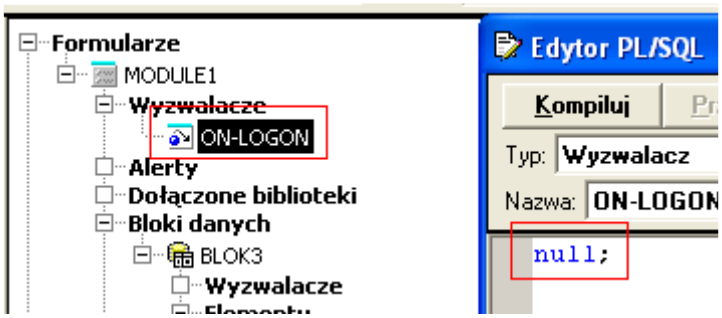
ZABEZPIECZANIE KONTA ROLĄ UAKTUWNIANĄ HASŁEM

Mechanizm ten stosuje się w celu uniemożliwienia dostępu do danych inaczej niż za pomocą formularzy (np. z poziomu SQLPLUS)

Utworzyć rolę ROLA.
Zabezpieczyć rolę hasłem HASLO. Można przyjąć ROLA = DBA (tak, żeby przyznanie roli dawało prawo do działania na obiektach KON)
Utworzyć użytkownika UZYTKOWNIK z hasłem HASLO2 (Role: CONNECT, ROLA)
ALTER USER UZYTKOWNIK DEFAULT ROLE ALL EXCEPT ROLA;

UAKTYWNIANIE ROLI ODBYWA SIĘ ZA POMOCĄ POLECENIA:
SET ROLE ROLA IDENTIFIED BY HASŁO

FORMULARZ BEZ LOGOWANIA DO BAZY



CZEGO W ORACLE NIE MA

TYPU LOGICZNEGO W BAZIE DANYCH (W FORMSACH, REPORTSACH JEST - BOOLEAN)

BUGI

W FORMSACH NIE MOŻNA UTWORZYĆ ZAPYTANIA: SELECT * FROM TABELA WHERE POLE IN (**:LISTA_WARTOŚCI**). ROZWIĄZANIE: TABELA TYMCZASOWA POSTACI (ID, WARTOSC, GDZIE: ID TO ID_SESJI BRANE Z SEKWENCJI, ŻEBY ZACHOWAĆ PROAWIDŁOWE DZIAŁANIE W SIECI, WARTOŚĆ TO WARTOŚĆ) ŁADOWANA WARTOŚCIAMI

DESIGNER

JAK GENEROWAĆ FORMULARZ Z TZW. SMARTBAR:
ZESTAW PREFERENCJI: FORM/MenuAttachment/ NameOfMenuModuleIfNotImplicit:
DEFAULT&SMARTBAR

Zostanie wygenerowany moduł z ustawieniami: MODUL.MENU = DEFAULT&SMARTBAR
Żadnych ustawień SMARTBAR nie można zmieniać

PYTANIA:

```
rg_id := Create_Group_From_Query('MY_QRY_GROUP',
        'SELECT ENAME,EMPNO,SAL FROM EMP ORDER BY SAL DESC');
  status := Populate_Group( rg_id );
```

MOŻNA PO TYM ŁAZIĆ ?

- Jak wypełnić blok niebazowy rezultatem polecenia CREATE_GROUP_FROM_QUERY

Jak sobie radzisz z problemem ? cbp, colombo – ta sama nazwa olb.

QUERY_PARAMETER ?