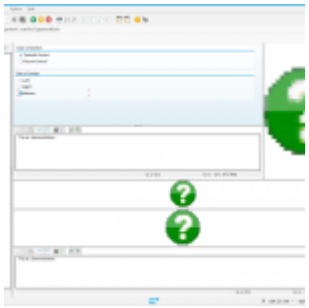


GUI-Designer „guidrasil“

Vor langer Zeit habe ich mich mit der automatischen und generischen Erzeugung und Verwaltung von *SAPGUI-Controls* beschäftigt.

Was sind Controls?

Controls sind ActiveX-Komponenten (auch OCX-Komponenten genannt), die im SAPGUI verwendet werden können und mit diesem ausgeliefert werden. Diese Windows-Komponenten werden über entsprechende Klassen im SAP angesprochen und erzeugt. Die Erzeugung erfolgt in der Regel ähnlich:

1. CREATE OBJECT <object reference>
2. <object reference>-SET_....
3. Manche Controls benötigen noch ein explizites DISPLAY.

Die typischen GUI-Controls sind:

- CL_GUI_ALV_GRID
- CL_GUI_TEXTEDIT
- CL_GUI_PICTURE
- CL_GUI_CALENDAR
- CL_GUI_HTML_VIEWER
- CL_GUI_SIMPLE_TREE
- CL_GUI_COLUMN_TREE
- CL_GUI_LIST_TREE

Controls benötigen einen Container, in dem sie platziert werden können. Lustiger Weise erben die Container-Klassen von der gleichen Klasse wie die Controls selber: CL_GUI_CONTROL. Die Container erben dann alle von CL_GUI_CONTAINER:

- CL_GUI_DOCKING_CONTAINER
- CL_GUI_CUSTOM_CONTAINER
- CL_GUI_DIALOGBOX_CONTAINER

Eine Sonderstellung nehmen die Splitter-Controls ein, denn sie stellen ebenfalls wieder Container zur Verfügung:

- CL_GUI_SPLITTER_CONTAINER
- CL_GUI_EASY_SPLITTER_CONTAINER

Programmierung von Controls

Eine typische Programmierung sieht wie folgt aus:

- Erzeuge einen Container
- Erzeuge das Control in diesem Container
- Setze Eigenschaften des Controls

In diesem Demoprogramm zeige ich kurz, wie ein Textedit-Control in einem Docking-Container aufgebaut wird.

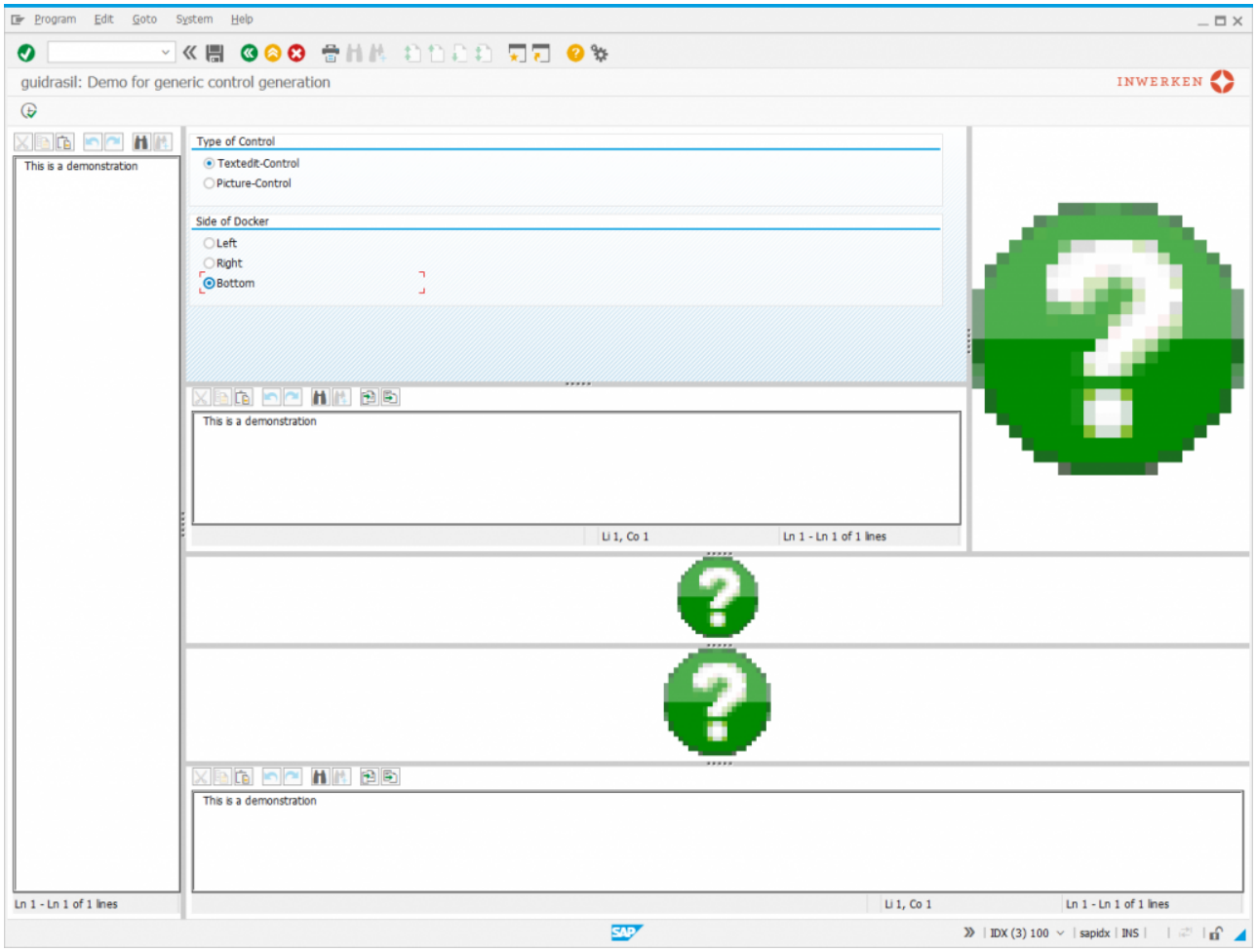
Setzen von Eigenschaften

Die Ansteuerung der Controls ist natürlich immer unterschiedlich, da sich die Control unterscheiden. Ein Picture-Control ist nun mal immer read-only, Ein Textedit-Control nicht. Das ist auch genau das Problem: Wenn ich ein Control häufig verwende, dann kenne ich die Eigenschaften und notwendigen Attribute. Wenn nicht, dann muss ich suchen. Zudem ist die Aktivierung von Attributen manchmal per BOOLEAN notwendig (X und space) und manchmal verlangt das Control „1“ und „0“.

Idee eines GUI-Designers

Da die Controls alle von der Klasse CL_GUI_CONTROL abstammen, ist es möglich jedes Control über eine generelle Methodenschnittstelle zu erzeugen. Ich kann also einer Methode irgend einen Container übergeben (egal, ob Docking-Container, Custom-Container oder Dialogbox) und das Control wieder zurück bekommen. Oder ich kann ein Control erzeugen und es in einer Tabelle speichern.

Das folgende Beispielprogramm macht genau das: Durch die Einstellungen auf dem Selektionsbildschirm wird definiert, welche Art von Control auf welcher Seite angedockt werden soll. Mit <ENTER> werden der Container sowie das Control erzeugt und in einer internen Tabelle abgelegt:



Demoprogramm

```
REPORT zguidrsail_demo_generic_ctrl.

SELECTION-SCREEN BEGIN OF BLOCK ctrl WITH FRAME TITLE TEXT-ctl.
PARAMETERS p_text RADIOBUTTON GROUP ctrl DEFAULT 'X'.
PARAMETERS p_icon RADIOBUTTON GROUP ctrl.
SELECTION-SCREEN END OF BLOCK ctrl.

SELECTION-SCREEN BEGIN OF BLOCK side WITH FRAME TITLE TEXT-sid.
PARAMETERS p_left RADIOBUTTON GROUP side DEFAULT 'X'.
PARAMETERS p_rigt RADIOBUTTON GROUP side.
PARAMETERS p_botm RADIOBUTTON GROUP side.
SELECTION-SCREEN END OF BLOCK side.

CLASS ctrl_demo DEFINITION.
  PUBLIC SECTION.
    METHODS add_text
      IMPORTING
        side TYPE i.
    METHODS add_icon
      IMPORTING
        side TYPE i.
  PROTECTED SECTION.
    TYPES: BEGIN OF ts_object,
```

```
            container TYPE REF TO cl_gui_container,
            control   TYPE REF TO cl_gui_control,
          END OF ts_object.

    DATA objects TYPE STANDARD TABLE OF ts_object.
    METHODS append_control
      IMPORTING
        container TYPE REF TO cl_gui_container
        control   TYPE REF TO cl_gui_control.

ENDCLASS.

CLASS ctrl_demo IMPLEMENTATION.
  METHOD add_text.
    DATA(parent) = NEW cl_gui_docking_container( side = side ratio = 20 ).
    DATA(textedit) = NEW cl_gui_textedit( parent = parent ).
    textedit->set_text_as_stream( VALUE texttab( ( tdline = `This is a demons
tration` ) ) ).
    append_control( container = parent control = textedit ).
  ENDMETHOD.
  METHOD add_icon.
    DATA(parent) = NEW cl_gui_docking_container( side = side ratio = 20 ).
    DATA(icon) = NEW cl_gui_picture( parent = parent ).
    icon->load_picture_from_sap_icons( icon_message_question ).
    icon->set_display_mode( cl_gui_picture=>display_mode_fit_center ).
    append_control( container = parent control = icon ).
  ENDMETHOD.
  METHOD append_control.
    APPEND VALUE #( container = container control = control ) TO objects.
  ENDMETHOD.
ENDCLASS.

INITIALIZATION.
  DATA(demo) = NEW ctrl_demo( ).

AT SELECTION-SCREEN.

  CASE 'X'.
    WHEN p_left.
      DATA(side) = cl_gui_docking_container=>dock_at_left.
    WHEN p_rigt.
      side = cl_gui_docking_container=>dock_at_right.
    WHEN p_botm.
      side = cl_gui_docking_container=>dock_at_bottom.
  ENDCASE.

  CASE 'X'.
    WHEN p_text.
      demo->add_text( side = side ).
    WHEN p_icon.
      demo->add_icon( side = side ).
```

```
  ENDCASE.
```

Dynamische Verwaltung

Da ich nun alle erzeugten Container und Controls in einer Tabelle habe, kann ich auch auf die Objekte und deren Eigenschaften zugreifen. Ich könnte zum Beispiel die Tabelle durchgehen und fragen: Ist im Feld CONTAINER ein Objekt der Klasse CL_GUI_DOCKING_CONTAINER? Wenn ja, frage ich das Control nach seinen wichtigen Eigenschaften: RATIO und SIDE:

```
IF itab-container IS INSTANCE OF cl_gui_docking_container.
  DATA dock TYPE REF TO cl_gui_docking_container.
  dock ?= itab-container.
  DATA(side) = dock->get_docking_side( ).
  dock->get_ratio( ratio = DATA(ratio) ).
ENDIF.
```

Auf diese Weise könnte ich mir alle wichtigen Eigenschaften eines Controls beschaffen und speichern.

Dynamische Erzeugung

Mit Hilfe von RTTI (Run Time Type Information) in Form der Klasse CL_ABAP_TYPEDESCR kann ich sogar den Klassennamen des Objektes ermitteln:

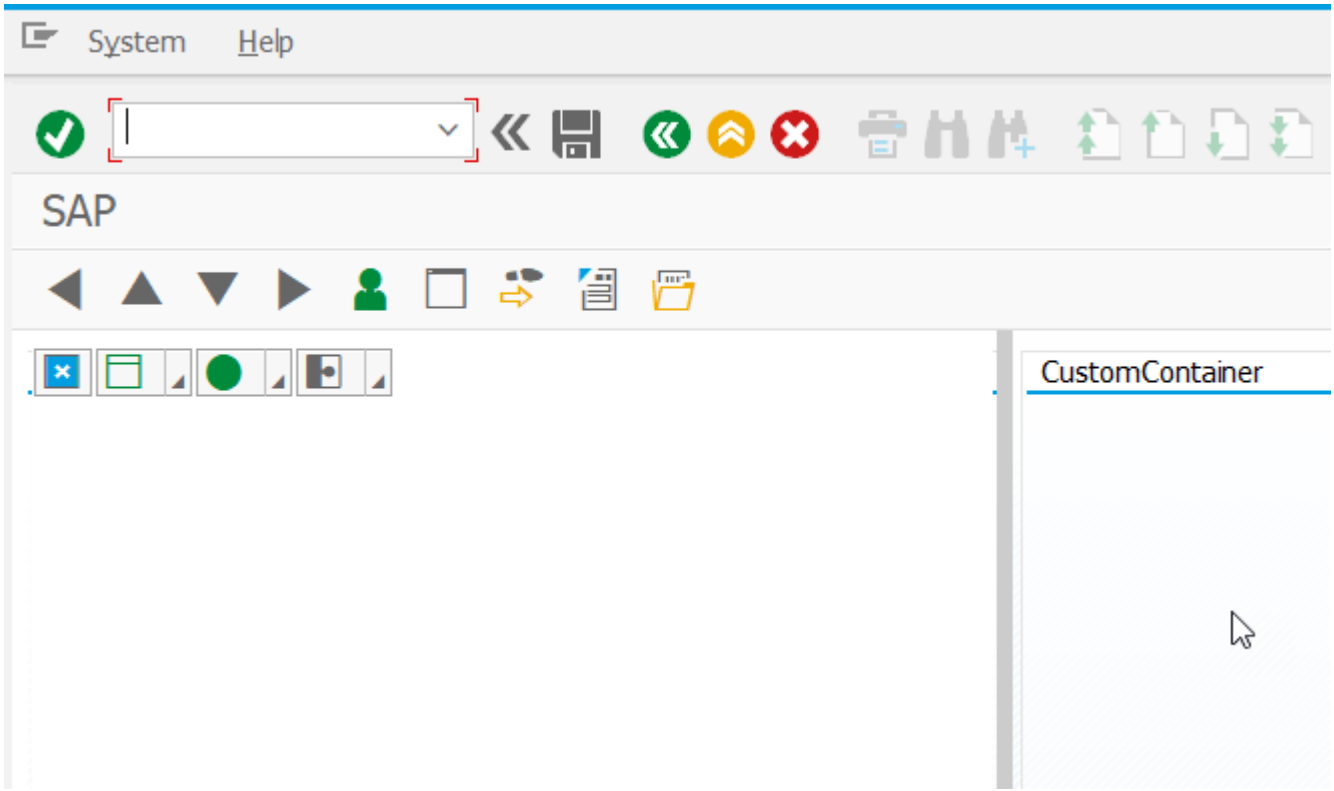
DATA(clsnam) = cl_abap_typedescr=>describe_by_object_ref(itab-
container)->get_relative_name( ).
Wenn ich diesen habe, dann ich das Objekt auch dynamisch erzeugen:

```
DATA: container TYPE REF TO cl_gui_container,
      exc_ref TYPE REF TO cx_root.

DATA: ptab TYPE abap_parmbind_tab.

ptab = VALUE #(
                ( name  = 'SIDE'
                  kind  = cl_abap_objectdescr=>exporting
                  value = REF #( side ) )
                ( name  = 'RATIO'
                  kind  = cl_abap_objectdescr=>exporting
                  value = REF #( ratio ) ) ).

TRY.
    CREATE OBJECT container TYPE (clsnam)
      PARAMETER-TABLE ptab.
  CATCH cx_sy_create_object_error INTO exc_ref.
    MESSAGE exc_ref->get_text( ) TYPE 'I'.
ENDTRY.
```

Eine dynamische Erzeugung ist jedoch gar nicht notwendig, denn ich kenne ja den Klassennamen und kann die Erzeugung wiederum an eine Erbauer-Klasse auslagern.

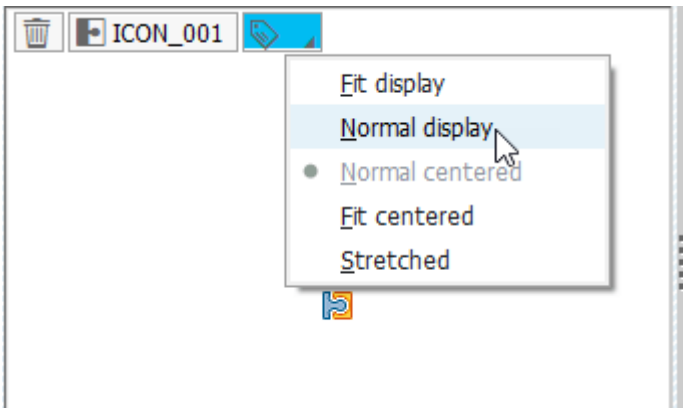
guidrasil

Nach dem oben beschriebenen Prinzip funktioniert der GUI-Designer ungefähr. Eine wichtige Eigenschaft ist jedoch, dass man erst auswählen muss, auf welcher Seite man einen Docking-Container erstellen möchte. In diesem Docking-Container wird dann ein Splitter erzeugt, der oben eine Toolbar integriert und unten erneut einen leeren Container anzeigt.



In der Toolbar kann man dann die gewünschten Controls oder Splitter-Container auswählen. Der GUI-Designer merkt sich, welche Objekte an welcher Stelle erzeugt werden. Das Erzeugen der einzelnen Controls sowie das Speichern der unterstützten Eigenschaften übernimmt die Erbauer-Klasse, die es für jedes Control gibt.

Zusätzlich stellt die Erbauer-Klasse auch noch einen Dialog zur Verfügung, in dem die Eigenschaften des Control eingestellt werden können.



I’d rather write code that writes code than write code

Eine weitere Eigenschaft der Erbauer-Klasse ist, dass jede Erbauer-Klasse ja genau weiß, wie das eigene Control erzeugt werden muss. Das heißt, es kann auch Code zur Verfügung stellen, der für die Erzeugung des Controls notwendig ist.

Der GUI-Designer weiß genau, welche Controls in welcher Reihenfolge erzeugt werden müssen. Der Designer muss also nur noch jedes Control nach dem Erzeugungscode fragen…

abapGit

Der GUI-Designer guidrasil ist verfügbar per abapGit auf Github:

https://github.com/tricktresor/guidrasil