Former Member

September 29, 2017 | 4 minute read

Dynamic Programming in ABAP – Part 3 – An Example – ABAP RTTS

5 6 19,564

Hi,

In my last blog I explained about the significance of field symbol and data references in dynamic programming.

https://blogs.sap.com/2017/09/05/dynamic-programming-in-abap-part-1-introduction-to-field-symbols/

https://blogs.sap.com/2017/09/11/dynamic-programming-in-abap-part-2-introduction-to-data-reference/

Now here we will see one example of dynamic programming approach and also a brief introduction to ABAP RTTS.

Follow

Like

RSS Feed

ABAP **Runtime Type Services (RTTS)** consists of two components:

- **Runtime Type Identification (RTTI)** – Provides the methods to get the type definition of data objects at runtime.
- **Runtime Type Creation (RTTC)** – Provides the methods to create the data objects at runtime with any type definition.

Basically, ABAP RTTS provides a set of classes, whose methods can be used for runtime type identification and runtime type creation. To know more about ABAP RTTS you can follow below link:

https://wiki.scn.sap.com/wiki/pages/viewpage.action?pageId=42965

An example of dynamic programming:

Requirement: As an ABAP developer, very often we get the situation where we need to write data from an internal table to a file on application server.

Solution: We will build one class having a method which will take any internal table as input and write its content in a file on application server.

Class Definition:

```
CLASS cl_appserver_writer DEFINITION.
  PUBLIC SECTION.
    CLASS-METHODS: write IMPORTING
                           iv_filename  TYPE string
                           it_data      TYPE ANY TABLE
                           write_header TYPE abap_bool DEFAULT space
                         EXPORTING
                           ev_message   TYPE string.
ENDCLASS.
```

Here importing parameter **it_data** is of **TYPE ANY TABLE** so that it can receive any internal table.

Class Implementation:

```
CLASS cl_appserver_writer IMPLEMENTATION.
  METHOD write.
    TYPES: BEGIN OF ty_comp_detail,
             name  TYPE abap_compname,
             descr TYPE scrtext_m,
           END OF ty_comp_detail.
    DATA: lo_type_def    TYPE REF TO cl_abap_typedescr.
    DATA: lo_struct_def  TYPE REF TO cl_abap_structdescr.
    DATA: lo_table_def   TYPE REF TO cl_abap_tabledescr.
    DATA: lo_data_def    TYPE REF TO cl_abap_datadescr.
    DATA: lo_element_def TYPE REF TO cl_abap_elemdescr.
    DATA: lt_components  TYPE abap_compdescr_tab.
    DATA: wa_components  LIKE LINE OF lt_components.
    DATA: lv_str         TYPE string.
    DATA: lv_filerow     TYPE string.
    DATA: lv_counter     TYPE i VALUE 0.
    DATA: lw_field_info  TYPE dfies.
    DATA: ls_comp_detail TYPE ty_comp_detail.
    DATA: lt_comp_detail TYPE TABLE OF ty_comp_detail.

    FIELD-SYMBOLS: <row> TYPE any.
    FIELD-SYMBOLS: <field_value> TYPE any.

* Using RTTS to get the runtime type information of the internal table
    lo_type_def  = cl_abap_tabledescr=>describe_by_data( it_data ).
    lo_table_def ?= lo_type_def.
    lo_data_def = lo_table_def->get_table_line_type( ).
    lo_struct_def ?= lo_data_def.

* Get the components of the structure
    lt_components = lo_struct_def->components.
```

Follow

Like

RSS Feed

```
    CLEAR: lo_data_def.

* If the WRITE_HEADER is ABAP_TRUE then fetch the label
* of data element associated to each component of the
* line type structure of internal table, if no data element
* is associated then use component name as the header text
    IF write_header EQ abap_true.
      LOOP AT lt_components INTO wa_components.
        lo_data_def = lo_struct_def->get_component_type( wa_components-name ).
        lo_element_def ?= lo_data_def.
        lw_field_info = lo_element_def->get_ddic_field( ).
        ls_comp_detail-name = lw_field_info-rollname.  "Get the data element name

* Calling FM to get data element text
        CALL FUNCTION 'WCGW_DATA_ELEMENT_TEXT_GET'
          EXPORTING
            i_data_element = lw_field_info-rollname
            i_language     = sy-langu
          IMPORTING
            e_scrtext_m    = ls_comp_detail-descr
          EXCEPTIONS
            error          = 1.
        IF ls_comp_detail-descr IS INITIAL.
          ls_comp_detail-descr = wa_components-name.
        ENDIF.
        APPEND ls_comp_detail TO lt_comp_detail.
        CLEAR: ls_comp_detail.
      ENDLOOP.
    ENDIF.


    OPEN DATASET iv_filename FOR OUTPUT IN TEXT MODE ENCODING DEFAULT.
    IF sy-subrc EQ 0.
* Writing header text for each column separated by comma
      IF write_header EQ abap_true.
```

Follow

Like

RSS Feed

```
        LOOP AT lt_comp_detail INTO ls_comp_detail.
          lv_counter = lv_counter + 1.
          IF lv_counter EQ 1.
            lv_filerow = ls_comp_detail-descr.
          ELSE.
            CONCATENATE lv_filerow ',' ls_comp_detail-descr INTO lv_filerow.
          ENDIF.
        ENDLOOP.
        TRANSFER lv_filerow TO iv_filename.
        CLEAR: lv_filerow, lv_counter.
      ENDIF.

* Writing internal table content separated by comma
      LOOP AT it_data ASSIGNING <row>.
        LOOP AT lt_components INTO wa_components.
          lv_counter = lv_counter + 1.
          ASSIGN COMPONENT wa_components-name OF STRUCTURE <row> TO <field_value>.
          IF <field_value> IS ASSIGNED.
            lv_str = <field_value>.
            IF lv_counter EQ 1.
              lv_filerow = lv_str.
            ELSE.
              CONCATENATE lv_filerow ',' lv_str INTO lv_filerow.
            ENDIF.
            UNASSIGN <field_value>.
          ENDIF.
        ENDLOOP.
        TRANSFER lv_filerow TO iv_filename.
        CLEAR: lv_filerow, lv_counter.
      ENDLOOP.
      CLOSE DATASET iv_filename.
      ev_message = 'Success'.
    ELSE.
      ev_message = 'Failure'.
    ENDIF.
```

```
  ENDMETHOD.
ENDCLASS.
```

Here the classes **CL_ABAP_*DESCR** are provided by the ABAP RTTS and used to get the type definition of data objects at runtime. Also we have extracted the data element name of each component of line type structure of internal table **it_data** using RTTS classes. Then we fetched the data element label using the FM **WCGW_DATA_ELEMENT_TEXT_GET**. This label is used to write the header for each column of internal table **it_data** if WRITE_HEADER parameter of class is provided with ABAP_TRUE.

Using the Class – The above designed class can be used as:

```
DATA: lt_data  TYPE STANDARD TABLE OF mara.
  DATA: lv_filename TYPE string.
  DATA: lv_message  TYPE string.

  SELECT * FROM mara INTO TABLE lt_data UP TO 5 ROWS.

  cl_appserver_writer=>write(
    EXPORTING
      iv_filename  = 'D:\usr\sap\testdata.csv'
      it_data      = lt_data
      write_header = abap_true
    IMPORTING
      ev_message   = lv_message
  ).

  WRITE: / lv_message.
```

Here we are passing one internal table of structure MARA to the class, and subsequently its content will be written on application server as comma separated values. However, we can pass internal table of any structure. This file can also be downloaded from application server to an excel spreadsheet.

Follow

So this is how field symbol, data reference, generic data type, RTTS helps in dynamic programming approach.

Like

The complete code for this scenario can be downloaded from:

RSS Feed

s: :h om/rkgupta94/ABAP-Development

Alert Moderator

Assigned Tags

ABAP Development

SAP NetWeaver Application Server for ABAP

abap

application

dynamic

example

identification

View more...

Similar Blog Posts

1000+ Pages full of ABAP

By Horst Keller Mar 20, 2007

Something's Coming

By Horst Keller Jan 15, 2007

Ten practices to make your ABAP developments optimized, informative, modern and quality robust

By Himanshu Sah Jan 28, 2021

Follow

Related Questions

Like

Toolbar Icons for RTTS output

By George Laing Feb 01, 2015

RSS Feed

How to create Dynamic class in abap programming.

By Ajay Rajareddy Aug 04, 2018

ABAP and Linear Programming

By Former Member Feb 21, 2008

Join the Conversation

SAP TechEd

Tune in for tech talk. Stay for inspiration. Upskill your future.

SAP BTP Learning Group

SAP Business Technology Platform Learning Journeys.

Coffee Corner

Join the new Coffee Corner Discussion Group.

5 Comments

Follow

You must be Logged on to comment or reply to a post.

Like

Jelena Perfiljeva
October 2, 2017 at 5:21 pm

Word of caution: I'm guessing one could run into a problem using CONCATENATE with some field types.

Also I stumbled upon the class CL_RSDA_CSV_CONVERTER with helpful methods CSV_TO_STRUCTURE and STRUCTURE_TO_CSV that handles comma separation much better than brutal CONCATENATE. E.g. it also accounts for the commas present in the data, in which case the field must be wrapped in the quotation marks ("). Not sure if there are even better methods for this, our system is rather old.

Thanks for sharing but you might want to work on the examples a bit more, going forward, especially if this is targeted towards the beginners. IMHO this particular example doesn't take full advantage of the presented functionality and if you're not doing that it could've just been much simpler (better for beginners).

"File can also be downloaded from application server to an excel spreadsheet" - could be a bit misleading. The file can be obtained from the application server but it'll still be a CSV file and what one does with it is up to them. Yes, Excel can read a CSV file but it's a separate process. Just to be clear. (We need to mind the potential audience in the "newbie" blogs.)

Thank you!

Like 3 | Share

Former Member | Blog Post Author
October 3, 2017 at 2:59 am

Thank you Jelena for your feedback and providing a better way for structure to CSV conversion.

Regards,

Rahul

Like 0 | Share

Like

Michelle Crapo

October 2, 2017 at 7:46 pm

Rahul,

Nice change from the last blogs.   Of course I agree with Jelena on the technical side. Maybe a revision would be a good idea.   But it was great to show how it was used instead of just the definition!

Keep going - each step is getting you closer to that perfect blog!

Michelle

Like 1 | Share

Former Member | Blog Post Author

October 3, 2017 at 3:01 am

Thank you Michelle for your feedback and motivation.

Regards,

Rahul

Like 1 | Share

Luc VANROBAYS
March 5, 2019 at 3:52 pm

Hello,

I changed as below in order it to work in system where FM WCGW_DATA_ELEMENT_TEXT_GET doesn't exist

```
* Calling FM to get data element text
        CALL FUNCTION 'WCGW_DATA_ELEMENT_TEXT_GET'
          EXPORTING
            i_data_element = lw_field_info-rollname
            i_language     = sy-langu
          IMPORTING
            e_scrtext_m    = ls_comp_detail-descr
          EXCEPTIONS
            error          = 1.
```

Substituted:

```
* Select on DD04VT instead of Calling FM (when not exists) to get data element text
 SELECT SINGLE SCRTEXT_M
 INTO ls_comp_detail-descr
 FROM DD04VVT
 WHERE DDLANGUAGE = 'EN'
 AND ROLLNAME = lw_field_info-rollname.
*        CALL FUNCTION 'WCGW_DATA_ELEMENT_TEXT_GET'
*          EXPORTING
*            i_data_element = lw_field_info-rollname
*            i_language     = sy-langu
*          IMPORTING
*            e_scrtext_m    = ls_comp_detail-descr
*          EXCEPTIONS
*            error          = 1.
```